Graph Based Reference Genomes

Knut Dagestad Rand

May 7, 2019

Thesis submitted for the degree of Philosophiæ Doctor

Contents

Chapter 1

List of papers

1.0.1 Paper I: Coordinates and Intervals in Graph Based Reference Genomes

Published in *BMC Bioinformatics*

1.0.2 Paper II: Graph Peak Caller: Calling ChIP-seq peaks on graph-based reference genomes

Published in *PLOS Computational Biology*

1.0.3 Paper III: Assessing graph-based read mappers against a novel baseline approach highlights strengths and weaknesses of the current generation of methods

Submitted to *Genome Biology*

1.0.4 Paper IV: Beware the Jaccard: the choice of metric is important and non-trivial in genomic colocalisation analysis

Submitted

Chapter 2

Introduction

The topic of this thesis is reference genomes, and in particular graph-based reference genomes. In four papers, various aspects of reference genomes are investigated. How sequenced reads are mapped to the reference genome, how mapped reads can be used in downstream analysis to deduce biochemical properties of DNA, how results from such analysis can be represented as coordinates and intervals on the reference genome, and how such coordinates and intervals can be used to investigate interplay between different biochemical processes. Three of the papers are focused on how these processes are performed using a graph based reference genomes.

In the next chapter, brief introductions are given to themes needed to give context to the use of reference genomes in bioinformatics. Starting out with an introduction to molecular biology in Section 3.1, Section 3.2 explains the process of DNA-sequencing which is how the observed data used in this thesis was obtained. In Section 3.3 the role of reference genomes is discussed, followed by two sections (3.5, 3.6) on alignment and mapping which introduces some technical aspects of one of the primary functions of reference genomes. Section 3.7 gives some background information on peak calling which is the theme of Paper II. Together this background should provide info on what reference genomes are and how they are used, as well as a motivation for using graph based reference genomes.

Chapter 3

Background

3.1 Molecular Biology

3.1.1 DNA

At the heart of molecular biology is *deoxyribonucleic acid*, DNA. DNA-molecules consist of pairs of four different *nucleotides* linked together to form a double stranded helix. The four different nucleotides are often represented by the letters A (*adenin*), C (*cytosin*), T (*thymin*) and G (*guanin*), forming an alphabet of nucleotides $\Gamma_{DNA} = \{A, C, T, G\}$. Each nucleotide can only be paired with one of the others, such that each base-pair in the helix is either (A-T) or (G-C) or opposite. The DNA in a cell is typically organized into long DNA molecules called *chromosomes*.

The most important function of DNA is to serve as a template for the synthesis of proteins, which in turn are responsible for a wide range of bio-molecular functions. This is a two-step process where DNA is first *transcribed* to RNA, molecules similar to DNA but single stranded and with *thymin* replaced by *uracil* (U). The resulting RNA-sequence can in turn be translated to proteins by mapping triplets of ribonucleotides to amino acids, which are the building blocks of proteins.

Sequences of DNA that are transcribed to RNA that either have a function in themselves, or are in turn translated to proteins, are called *genes*. Genes constitute only a minor proportion of human DNA. The remaining DNA can be important due to the biochemical properties of the DNA itself.

3.1.2 Replication and Mutation

In addition to serving as template for RNA molecules, DNA can also be replicated. This leads to inheritance both on cellular and organismal level. In normal cell division, the chromosomes are replicated such that each of the two resulting cells have a copy of the DNA from the original cell. Furthermore, germ line cells are copied and transfers half of the organisms chromosomes to an offspring.

However, this DNA-replication is not always accurate. Errors can occur leading to a copy of the DNA molecule which is not identical with the original. Most common are the substitution of a single nucleotide, insertion of a small DNA sequence, or deletion of a small subsequence. Such errors lead to difference between cells within an organism, or difference between individuals.

Difference in the DNA-sequence between two individuals can lead to a change in the transcribed RNA sequence and further in the sequence, and thereby the form and function, of the expressed protein. It can also lead to less drastic changes such as changes in the RNA-structure or the shape of the DNA molecule itself. In this way genomic variation can determine differences between specimens of the same species and also differences between the species themselves.

3.1.3 Epigenomics

Difference in DNA-sequence can explain phenotypic difference between individuals, but fails to explain the difference between the different cells within an organism. Within an organism, the cells contain mostly the same DNA-sequences, but exhibit vastly different phenotypes.

The reason for this difference is mostly attributed to differences in expression levels of genes. Much attention has been devoted in recent years to the mechanisms determining gene expression levels. Two high-level consortia, ENCODE [5] and Roadmap Epigenomics [1], have systematically conducted experiments geared towards understanding some of these factors, including: 3D-configuration of the chromosomes, accessiblity of the DNA in different regions, transcription factor binding sites, and methylation patterns across the genome. Among these, this thesis is concentrated on transcription factor binding sites.

Transcription factors are proteins that bind to DNA, with an affinity for certain DNA-sequences. These proteins signal to other proteins that a gene their vicinity should be transcribed, thus affecting expression levels of genes. As such the presence or absence of transcription factors binding to a binding site can affect

the phenotype of a cell. Also, since the binding of transcription factors is dependent on the DNA-sequence, a mutation in the DNA-sequence in a binding site can affect the binding of the protein and thus the expression of a nearby gene.

3.2   Sequencing

The fact that the sequence of nucleotides in the DNA has direct links to biological function has made determining such sequences as well as possible an important goal in molecular biology. The process of determining the sequence of nucleotides is referred to as DNA-*sequencing*. The first widely used method for DNA-sequencing was Sanger Sequencing [32], developed in 1976. The throughput from Sanger sequencing is pretty low since it required measuring the weight of many nucleotides. In the beginning of the 2000s, several technologies were developed that were able to parallelize the process of sequencing and thus increased the throughput. Among these technologies, collectively called Next-Generation Sequencing (NGS), Illumina dye sequencing is the most commonly used, and will be described here.

3.2.1   Illumina Dye Sequencing

Illumina dye sequencing uses sequencing by synthesis, which is the same base methodology as Sanger sequencing. This method uses the property that given a single strand of DNA and the presence of DNA polymerase, nucleotides will sequentially bind to the strand according to the complementary pairings (A-T) and (C-G). This usually happens so fast that it is difficult to measure any point in this process. However, nucleotides can be adapted so that the synthesis is terminated after including the adapted nucleotide (chain termination). Sanger sequencing uses this by combining a small amount of adapted nucleotides with a larger amount of normal nucleotides, and then using this combined set for DNA synthesis. Illumina sequencing evolves this concept by also being able to reverse these adaptions so that synthesis can continue. Thus it can step by step synthesize one adapted nucleotide to the single DNA-strand, find out which nucleotide was included, then reverse the adaption and synthesize a new adapted nucleotide(see Figure 3.1).

The big breakthrough for this sequencing technology was that it could be performed massively in parallel and thereby generate much more sequencing data. The downside is that while Sanger sequencing could yield up to 700 base pairs long sequences, Illumina sequencing typically yields shorter reads (30-300 base pairs). This massive amount of relatively short reads is able to give great insights into biology by mapping them to reference genomes. Reference genomes are covered in Section 3.3, and the mapping process is covered in Section 3.6.

```
A  ··T   A  ·· T   A  ·· T   A  ·· T
|        |         |         |
C        C  ·· G   C  ·· G   C  ·· G
|        |         |         |
C        C         C  ··· G  C  ·· G
|        |         |         |
T        T         T         T     A
|        |         |         |
G        G         G         G
```
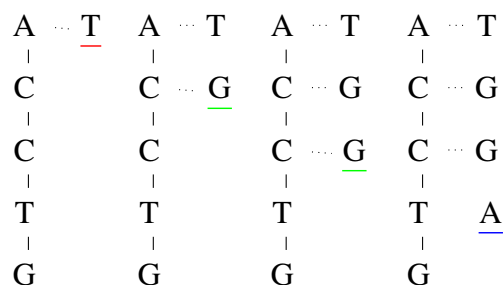
Figure 3.1: Sequencing of one DNA-chain in Illumina sequencing. For each step, an adapted nucleotide is added with a phosphorescent color, a picture is taken - capturing the color, and the adaption is reversed allowing for the next nucleotide to be bound. Each nucleotide (A, C, G, T) has a different color, so the sequence of colors in the images taken can be used to determine the sequence of nucleotides.

The main error source of Illumina sequencing is that a non-adapted nucleotide can get included in one of the steps, whereupon it is possible for a new nucleotide to get included without the previous one being captured. To alleviate this, and to obtain a stronger signal, Illumina performs the sequencing steps concurrently on many identical DNA templates. Even so, the number of out-of-sync strands grows with the length of the sequence and puts a practical limit on how many base pairs can be read with certainty.

Depending on which input DNA is provided, this technology can be used to answer a range of biological questions. One can input unfiltered DNA for sequencing and use the resulting reads to deduce as much of the whole genomic sequence as possible. One can also filter the DNA to contain mostly fragments from protein coding parts of the genome, in order to more efficiently find the DNA-sequence of these regions, from which variations can cause differences in protein structure and function.

In addition to deducing the DNA-sequence of a sample, NGS can also be used to deduce where in the genome biochemical processes occur, for instance the binding of proteins to the genomes. Of importance to this thesis are experiments used to find binding sites for transcription factors, described below.

3.2.2   ChIP-seq

*Ch*romatin *I*mmuno*p*resipitation followed by *seq*uencing (ChIP-seq) is a sequencing experiment that does not aim to determine the genetic variation of the sample,

but rather to give insight to where a protein of interest binds to the DNA.

In order to do this, the binding of the protein to DNA is strengthened in order to keep the bindings after the DNA is cut into fragments. Antibodies for the protein of interest are then used to retrieve DNA-fragments with the protein attached. These DNA-fragments can then be sequenced, yielding reads that tend to come from the vicinity of a binding site (Figure 3.2). The process of analyzing these reads is covered in Section 3.7.

3.2.3   Third Generation Sequencing

New technologies are continuously being developed, and a set of more recent technologies allows for sequencing of long DNA-molecules. This does however come at the price of much higher error rates than Illumina sequencing. Most notable is Oxford Nanopore [3], passing DNA-molecules through a molecule which emits a different electric current for each type of nucleotide, and PacBio [30], using nucleotide-dependent light waves emitted continuously while synthesizing. These techniques offer possibilities for answering biological questions where short read lengths are insufficient, but also new computational challenges for dealing with the high error rates and long read lengths.
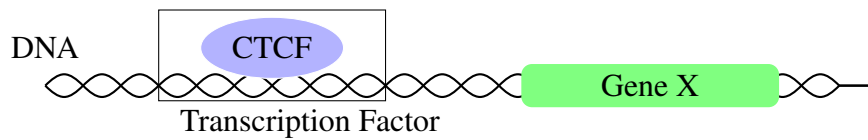
(a) Transcription factor binding to DNA, regulating the expression of Gene X

```
                                         CTCF
Cell1:    ACGTTCGTATATCGTAGCTACTCGAGCTGTAGTTTGATAGATAT
Cell2:    ACGTTCGTATATCGTAGCTACTCG....................
Cell3:    .......TATATCGTAGCTACTCGAGCTGTA.............
          ...........TCGTAGCTACTCGAGCTGTAGTTT.........
          ..GTTCGTATATCGTAGCTACTCGAG..................
          ...TTCGTATATCGTAGCTACTCGAGC.................
          ..................CTACTCGAGCTGTAGTTTGATAGA...
          ...................ACTCGAGCTGTAGTTTGATAGATA.
          ............GTAGCTACTCGAGCTGTAGTTTGA.......
```

(b) DNA Fragments obtained from ChIP different cells

```
                            CTCF
ACGTTCGTATATCGTAGCTACTCGAGCTGTAGTTTGATAGATAT
ACGTTCGTATATCGTAGCTACTCG....................
.......TATATCGTAGCTACTCGAGCTGTA.............
...........TCGTAGCTACTCGAGCTGTAGTTT.........
..GTTCGTATATCGTAGCTACTCGAG..................
...TTCGTATATCGTAGCTACTCGAGC.................
..................CTACTCGAGCTGTAGTTTGATAGA...
...................ACTCGAGCTGTAGTTTGATAGATA.
............GTAGCTACTCGAGCTGTAGTTTGA.......
                        ↓
ACGTTCGTAT, TACAGCTCGA, TCGTAGCTAC, GTTCGTATAT
GCTCGAGTAG , CTACTCGAGC, TATCTATCAA, TCAAACTACA
```

(c) One end of each fragment is sequenced

Figure 3.2: Figure showing the main steps of a CHiP-seq experiment. a) A protein of interest (CTCF) binds to the DNA; b) the DNA is cut up into fragments, from which the ones with proteins bound to it are kept; c) these fragments are then sequenced, yielding reads that are from the vicinity of protein binding sites.

Figure 3.3: Figure showing the locations of several genomic elements on the same coordinate system. Locations of genes (red), transcription factor binding sites (blue), and disease associated variants (green) from different sequencing experiments can all be compared and analyzed together. In this case two of the variants are located inside a known gene, and one is located in a predicted transcription factor binding site.

3.3 Reference Genomes

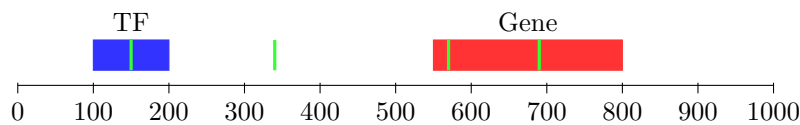When it can be assumed that the genomic DNA-sequences from individuals of the same population is highly similar, it is possible to use one DNA-sequence as a *reference genome* for the population. Such reference genomes serve two main purposes. They make the deduction of the DNA-sequence from a sample a simpler problem through mapping (see Section 3.6), and they make it possible to represent the outcome of a range of biological experiments as intervals or coordinates on the coordinate system induced by the reference genome.

3.3.1 Colocalization Analysis

The ability to represent genomic features as coordinates or intervals on a common coordinate system makes it possible to analyze the results of an experiment in light of previously accumulated data. For instance, predicted locations of genes in the human genome are available as sets of intervals from for instance [13]. Such sets can for instance be used to find the closest gene to a predicted transcription factor binding region obtained from a ChIP-seq experiment, thus obtaining a link to the phenotypic role of the transcription factor binding. Genome wide association studies [38] can be used to find genomic variants that are associated with a certain disease. The location of such variants can be represented in reference genome coordinates and thus be compared to the location of genomic functional elements. If a disease associated variant is located inside a protein coding gene, this indicates that the function of the protein can be linked to the disease. Similarly, if the variant is located in a predicted transcription factor binding site, this can give an indication that the gene closest to the binding site is relevant for the disease (see Figure 3.3).

Such analyses can be conducted on a genomic scale with the use of software like The Genomic Hyperbrowser [31] and others [33, 29]. This enables answering questions like whether variants associated with multiple sclerosis is overrepresented in accessible chromatin regions of immune cells [6]. Paper IV in this thesis discusses how the choice of similarity measures between sets of genomic intervals can affect the outcome of such genome wide colocalization analyses.

3.3.2 Graph Based References

Both mapping of reads and any subsequent analysis of the predicted genomic features is affected by the reference genome used. The more different the sample genome is from the reference, the less accurate the analyses become. In this light the inclusion of known common variants into the reference can be beneficial as the reference would then represent the population more accurately. Representing the reference as a sequence graph where different paths represents different variants is one way to achieve this [28]. The first three papers of this thesis discuss various facets of a change from linear reference sequences to reference sequence graphs; Paper I discuss the representation of genomic intervals on sequence graphs, Paper II introduces a method for analyzing ChIP-seq data on reference graphs and Paper III discusses aspects of mapping reads to a reference graph. The central concepts of mapping and aligning reads to a sequence graph are more thoroughly introduced in Sections 3.5 and 3.6.

3.4 Mathematical Framework

3.4.1 Notation

Sequences are important in this thesis as they are the means to represent DNA-sequences. We will here formally work with an alphabet $\Gamma_{DNA} = \{A, C, G, T\}$ and say that a *sequence* of length n over that alphabet is a tuple in the set Γ_{DNA}^n. The set of all sequences of any length over $\Gamma_{DNA} = \{A, C, G, T\}$ is represented by Γ_{DNA}^*. The length of a sequence $S \in \Gamma_{DNA}^*$ is notated $|S|$. The ith symbol in a sequence s is notated $S[i]$ and a substring of S starting at the ith symbol (inclusive) and ending at the jth symbol (exclusive) is denoted $S[i : j]$. A *prefix* of S, i.e. a substring of S starting from the first symbol, of length k is denoted $S[: k]$, while a *suffix* of S starting at the kth symbol is denoted $S[k :]$. A *kmer* of a sequence S is any substring of S of length k.

For convenience, a string of length n is sometimes represented as $S[: n]$ to convey the size of the string. Concatenation of two strings S, T are represented as $S * T$, while the concatenation of a symbol a to a string S is denoted Sa, and a string to a symbol as aS.

For some algorithms a # symbol is added to the start and/or a \$ symbol is added to the end of the string.

3.4.2 Graphs

A graph is a tuple $G = (V, E)$ of vertices and edges, where the edges are pairs of vertices $E \subset V^2$. We will here deal with directed graphs where we say that edge (v_1, v_2) is an edge from v_1 to v_2. An *path* is an alternating sequence of vertices and edges $(v_1, e_1, v_2, e_2, , , e_{n-1}, v_n)$ where $e_i = (v_i, v_{i+1})$. A *cycle* is a path $(v_1, e_1, v_2, e_2, , , e_{n-1}, v_n)$ where $v_1 = v_n$ and $n > 1$, i.e. a path that starts and ends at the same vertex. A *directed acyclic graph* (DAG) is a directed graph in which there are no cycles. We call the set of all paths starting at v_s and ending at v_e $\mathcal{P}(v_s, v_e)$.

Sequence Graphs

We define a *simple sequence graph* $SG = \{G, \ell\}$ over an alphabet Γ as a graph $G = \{V, E\}$ and a label function $\ell : V \to \Gamma$ labeling each vertex with a letter from Γ. We refer to the label of a path $\ell((v_1, e_1, v_2, , , e_{n-1}v_n))$ as the concatenation of the labels of its vertices $\ell(v_1) * \ell(v_2)...\ell(v_n)$. The sequence graphs we look at here

will have clearly defined start and end nodes, which will be denoted v_s and v_e. If this were not the case, dummy nodes ($\#, \$$) can be inserted to the graph to make the start and end unambiguous. The *language* recognized by a sequence graph $\mathcal{L}((G, \ell))$ is the label of each path going from the start node to the end node, i.e:

$$\mathcal{L}((G, \ell)) = \{\ell(p) \mid p \in \mathcal{P}(v_s, v_e)\}$$

3.5 Alignment

3.5.1 Edit Distance

Edit distance is a measure of how many edit operations are required to convert one sequence into another. Different edit distance measures exist, varying in the set of allowed operations. The most common is the *Levenshtein distance* [19], allowing single character substitutions, insertions and deletions.

Finding the edit distance between two sequences, and which set of edits this edit distance corresponds to, is of central importance in bioinformatics since it can give an estimate of how related the two sequences are and which mutations have separated them. The process of finding these estimates is called *sequence alignment* and is important for this thesis in two respects. Firstly, it is one of the earliest and most intuitive applications of sequence graphs, and secondly it is an important part of *mapping* which will be covered in the next section.

In the following we will refer to the edit distance between two sequences, S and T, as $D(S, T)$, meaning the Levenshtein distance unless specifically mentioned. The concepts discussed are however generalisable to other (weighted) edit distances by small changes.

The set of edits between two sequences can be represented with an *alignment block* where "-" symbols are inserted into the sequences to represent insertions and deletions (indels), (see Figure 3.4).

3.5.2 Pairwise Sequence Alignment

Finding the edit distance between two strings is easy if indels are not considered. It is merely the process of counting the number of mismatches between them. Indels introduce a dependency since an insertion at position i affects the pairings of all the symbols after position i. The number of meaningful combinations of insertions and deletions grow exponentially with sequence length, so exploring all of them is not a viable solution even for short sequences. The problem is however tractable since the best alignment of two sequences $S[: m], T[: n]$ is either the best alignment of $S[: m-1], T[: n]$ with an insertion at the end, or of $S[: m], T[: n-1]$ with a deletion at the end, or of $S[: n-1], T[: m-1]$ with a match or substitution at the end. Letting $d_{i,j} = D(S[: i], T[: j])$ and $m_{i,j}$ be zero if $S[i] = T[j]$ else 1,

we can write this as the recurrence relation:

$$d_{k,l} = min \begin{cases} d_{k-1,l-1} + m_{k-1,l-1} & \text{(match/substitution)} \\ d_{k-1,l} + 1 & \text{(insertion)} \\ d_{k,l-1} + 1 & \text{(deletion)} \end{cases}$$

The fact that the edit distance to a empty string is just the sequence length gives the initial conditions $d_{k,0} = k$, $d_{0,l} = l$. The Needleman-Wunch algorithm [27] uses dynamic programming to calculate the matrix of edit distances d_{kl}, where d_{mn} is the edit distance between S and T. In order to find the specific edits, a backtracking algorithm is used that starts at the $(i, j) = (m, n)$ corner and finds out which of the possible predecessors $(i-1, j), (i-1, j-1), (i, j-1)$ contributed to the (i, j) edit distance. Then repeating this until the $(0, 0)$ corner is reached. Each of these steps correspond to a column in the alignment block. Figure 3.4 details the Needleman Wunch algorithm. This algorithm can be adopted in a number of ways in order to solve related problems. Notably, by using an affine gap penalty one can reduce the cost of long indels compared to many small indels [10], or one can use positive scores for matches to be able to find subsequences that align well to each other [37].

An adaption relevant for this thesis, referred to as the *mapping* problem, is to find the a subsequence of one sequence R that minimizes the edit distance to another Q. I.e find $\min_{k,l}(D(R[k, l], Q)$. This can be done by changing just the initial conditions of the Needleman-Wunch algorithm, to remove the cost of gaps at the beginning and end of R. We set $d_{k0} = 0$ and start the backtracking algorithm at (m, l) where $l = \arg\min_i d_{mi}$. Figure 3.5 shows an example of this algorithm. This algorithm provides three results: the coordinate k, l of the substring of R most similar to Q, the edit distance from that substring to Q, and the set of edits contributing to the edit distance. In this way it solves in a exact, but slow way the problem of the next chapter, namely mapping a read Q to a reference genome R.

3.5.3 Sequence Graph Alignment

The framework used to align sequences extends naturally to acyclic sequence graphs [11, 18]. We define the alignment of a sequence S to a sequence graph G as the alignment of S to a sequence $T \in \mathcal{L}(G)$ in the language of G ($\mathcal{L}(G)$) which yields the lowest edit distance. Similarly, the alignment of two sequence graphs G, H, is the alignment of a sequence $S \in \mathcal{L}(G)$ to a sequence $T \in \mathcal{L}(G)$

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

```
       A    C    T    T    G    A
    0   -1   -2   -3   -4   -5   -6
A  -1    0   -1   -2   -3   -4   -5
C  -2   -1    0   -1   -2   -3   -4
G  -3   -2   -1   -1   -2   -2   -3
T  -4   -3   -2   -1   -1   -2   -3
T  -5   -4   -3   -2   -1   -2   -3
A  -6   -5   -4   -3   -2   -2   -2
C  -7   -6   -5   -4   -3   -3   -3
```

```
A  C  G  T  T  -  A  C
A  C  -  T  T  G  A  -
```

Figure 3.4: Figure showing the alignment of two sequences using Needleman-Wunch. Each cell in the matrix corresponds to the edit distance between prefixes of S and T. The red path shows the backtracking, resulting in an alignment block where each diagonal arrow gives a symbol from both sequences, while horizontal or vertical arrows give an insertion or deletion.

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

		A	C	T	T	G	A
	0	-1	-2	-3	-4	-5	-6
G	0	-1	-2	-3	-4	-4	-5
T	0	-1	-2	-2	-3	-4	-5
G	0	-1	-2	-3	-3	-3	-4
G	0	-1	-2	-3	-4	-3	-4
A	0	0	-1	-2	-3	-4	-3
C	0	-1	0	-1	-2	-3	-4
G	0	-1	-1	-1	-2	-2	-3
T	0	-1	-2	-1	-1	-2	-3
T	0	-1	-2	-2	-1	-2	-3
A	0	0	-1	-2	-2	-2	-2
C	0	-1	0	-1	-2	-3	-3
G	0	-1	-1	-1	-2	-2	-3
C	0	-1	-1	-2	-2	-3	-3
G	0	-1	-2	-2	-3	-2	-3
G	0	-1	-2	-3	-3	-3	-3
T	0	-1	-2	-2	-3	-4	-4

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

Figure 3.5: Figure showing the alignment of a query sequence Q to a reference sequence R, Each cell in the matrix corresponds to the edit distance between a prefix of S and a subsequence of R. The red path shows the backtracking, starting at the lowest value of the last column, resulting in an alignment block where each diagonal arrow gives a symbol from both sequences, while horizontal or vertical arrows give an insertion or deletion.

that yields the lowest edit distance.

If we let

$$d_{ij} = \min_{p_g \in \mathcal{P}(v_0, v_i), p_h \in \mathcal{P}(w_0, w_k)} (D(\ell(p_h), \ell(p_g)))$$

we get a recurrence relation:

$$d_{ij} = \min_{v_k \in e^- v_i, w_l \in e^- w_l} d^*(i, j, k, l)$$

$$d^*(i, j, k, l) = \min \begin{cases} d_{il} + 1 \\ d_{kj} + 1 \\ d_{kl} + m_{kl} \end{cases}$$

$$m_{ij} = 1 \text{ if } \ell(v_i) \neq \ell(w_j) \text{ else } 0$$

This is the same as for ordinary sequence alignment, except that all predecessor nodes of v_i and w_k have to be considered, not only $i - 1$ and $j - 1$ as in the linear case. If the graph is acyclic, then all the d_{ij} can be calculated using dynamic programming, without incurring any infinite loops.

An alignment between two sequences S, T can be represented by a sequence graph in a meaningful manner in that one can construct a sequence graph $AG(S, T)$ from the alignment of the sequences in such a way that

$$\forall(R \in \mathcal{L}(AG(S, T))) \, [D(S, R) + D(R, T) = D(S, T)].$$

For an optimal alignment, this means that all sequences recognized by the sequence graph have the property that the sum of the distance to the original sequences is as low as it can be. These sequences thus represent combinations of S and T that are natural estimates of an ancestor of the two sequences. Thus aligning a sequence S to an alignment graph $AG(T, R)$ can be seen as aligning S to the best fitting ancestor sequence of T and R Similarly, aligning two alignment graphs can be seen as finding the ancestors of two pairs of sequences that fit best together [12]. This is illustrated in figure 3.7.

The sequence graph alignment algorithm can also be adapted to find the substring of a sequence graph G_R that minimizes the edit distance to a sequence Q. This solves the problem of mapping a read to a graph, which will be discussed further in section 3.6.2.

− A − C − G − G − G
         T

− A − T − A − G

.   # → A → C   T → G → G → G

	#	A	C	T	G	G	G
#	0	0	-1	-2	-2	-3	-4
A	0	0	-1	-1	-2	-3	-3
T	-1	-1	-1	0	-1	-2	-2
A	-2	0	-1	-1	-1	-2	-2
G	-2	-1	-1	-1	-1	-1	-1

```
#   A   T   A   G
#   A   T   G   G
```
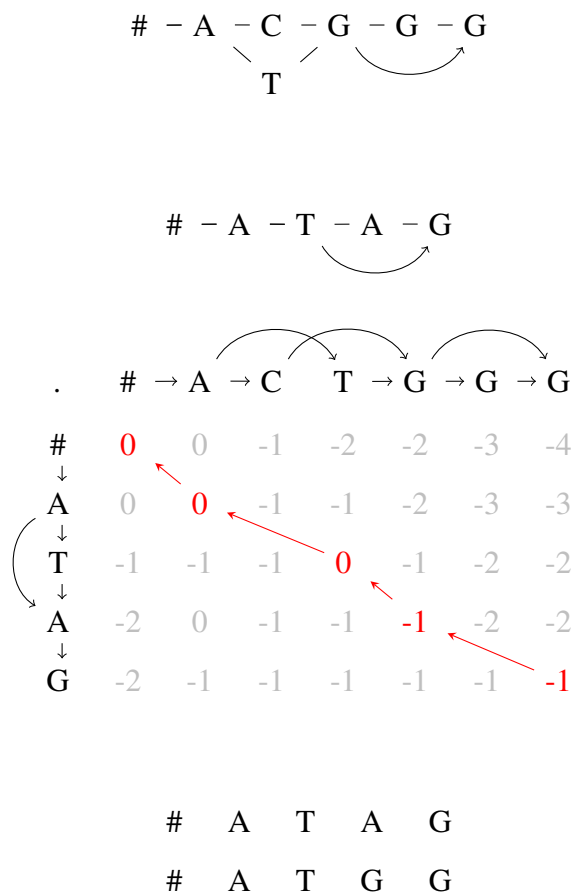
Figure 3.6: Figure showing the alignment of two sequence graphs. Marked in red is the path taken during backtracking. The result is the best alignment of a sequence from the language of each sequence graph. This alignment can again be represented as a sequence graph

X1: A → C → G → G → A → T → C → G → T → C → C → C → T

X2: C → G → G → A → T → C → T → T → C → C → C → T

```
                                        T
                                      ↗   ↘
X̂:  A → C → G → G → A → T → C → G → T → C → C → C → T
```

Y1: A → T → G → T → A → C → C → G → T → A → C → G → T

Y2: A → C → G → T → A → C → C → G → T → T → T → A → G → G → T

```
            C                                          G
          ↗   ↘                                      ↗   ↘
Ŷ:  A → T → G → T → A → C → C → G → T → T → T → A → C → G → T
```

```
                          T                    A → G
                        ↗   ↘                ↗       ↘
Ŝ:  A → C → G → G → A → C → C → G → T → C → C → G → T
```

Figure 3.7: Iterative sequence graph alignments of four sequences evolved in two generations from $S = ACGTACGTACGT$. The sequences are represented as linear sequence graphs (X1, X2, Y1, Y2) and pairwise alignment is performed on the closest pairs yielding two sequence graphs $(\hat{X}, \hat{Y})$. These two sequence graphs are then aligned to each other, yielding a sequence graph representing an alignment of the two closest paths in the graphs $(\hat{S})$. As seen the original sequence is in the language of the final sequence graph

3.6 Mapping

The alignment methods described in the previous section do not scale well to large sequences. Aligning a read $Q[: m]$ to a reference genome $R[: N]$ will require computing mN values. A NGS sequencing experiment typically yields hundreds of thousands of reads of length 50 which need to be located in the human reference genome of length 3 billions. This would need to calculate 10^{16} values, which is intractable even on modern computers.

This has led to much progress since the dawn of NGS in developing alignment methods that avoid the complexity of the exact dynamic programming methods. This has been achieved mainly by two approaches, creating searchable indexes of the reference genome and using heuristics to limit the search space of possible alignments.

The following is a brief description of the developments in this field, with focus on the aligner used in this thesis' projects: BWA-mem [21]. After this we will look at the methods used for graph alignment with focus on *vg* [9], which is also central in our work.

3.6.1 Linear Mapping

As in section 3.5.2, we define the mapping problem as finding the interval i_R for wich the edit distance $D(R[i_R], Q)$ is smallest, and the alignment $A(R[i_R], Q)$.

Indexing

An essential tool for quick alignment is to have a searchable index of the reference genome which is capable of returning sets of interval-pairs from the reference and query sequence for which the reference sequence is identical or near identical to the query sequence. The indexes used by different tools vary, but can mainly be divided into fixed-length (kmer) indexes [22] and variable-length indexes [17, 24]. Fixed length indexes typically use hash tables to store the location of a subset of the kmers in the reference sequence. Variable length indexes usually use a variation of the *full text minute index* (FM-index) [8], described below. The main idea covered here is the seed-and-extend paradigm. This idea consists of finding a set of exact matches to the reference sequence (seeds), and using these as anchors for using dynamic programming based alignment of the query. We let $EM(Q, R)$ be the set of all exact matches, represented by the tuples $\{(q, r, l) \mid Q[q : q + l] = R[r : r + 1]\}$. Since the dynamic programming methods

can still be computationally expensive, the goal is to make the seeds as small a subset of exact matches as possible, while still making sure that the true alignment covers one of the seeds.

FM-index

The FM-index uses a succinct representation of the suffix array [26] and Burrows-Wheeler transform [4] of the sequence in order to find exact matches of a query string S in $O(|S|)$ time (see Figure 3.8). Thus if a read has one or more exact matches in the reference sequence it can be found directly using the FM-index. For inexact matches, it can also be used to find an exact alignment of Q and R in approximately $O(|R|^{0.628}|Q|)$ time [15, 24]. Some algorithms also use the FM-index to find exact matches for permutations of the query sequence [16, 23], but for longer reads the search space gets too big when allowing for indels.

In order to handle indels in longer reads, the main methodology is to use the FM-index to find exact matches between substrings of the query and reference sequence, called seeds, and then aligning the reads using dynamic programing to intervals surrounding the seed matches on the reference sequence [17], often called the seed-and-extend paradigm.

One way of finding seeds is to find Maximal Exact Matches (MEM) [25]. Maximal Exact Matches are Exact Matches that cannot be extended in either direction, ie.

$$MEM(Q,R) = \{(q,r,l) \in EM \mid (q,r,l+1) \notin EM \wedge$$
$$(q-1,r-1,l+1) \notin EM\}$$

BWA-mem furthers this concept to SuperMaximal Exact Matches (SMEM) [20]. A SMEM is a MEM where the query interval cannot be extended in either direction and still yields an Exact Match anywhere on the reference 3.9.

$$SMEM(Q,R) = \{(q,r,l) \in EM \mid \nexists r^*[(q,r^*,l+1) \in EM \vee$$
$$(q-1,r^*,l+1) \in EM]\}$$

SMEMS are natural to use as seeds as they cover for each subsequence in Q the longest exact match in R. They are however vulnerable for spurious long matches hiding shorter exact matches. To account for this BWA-mem allows an option to split long SMEMs into shorter MEMs if they are longer than a certain

C T A G C T G C A T G
0 1 2 3 4 5 6 7 8 9 10

11	$	CTAGCTGCAT	G_0	0	0	0	0
2	A_0	GCTGCATG\$C	T_0	0	0	1	0
8	A_1	TG\$CTAGCTG	C_0	0	0	1	1
7	C_0	ATG\$CTAGCT	G_1	0	1	1	1
0	C_1	TAGCTGCATG	$	0	1	2	1
4	C_2	TGCATG\$CTA	G_2	0	1	2	1
10	G_0	\$CTAGCTGCA	T_1	0	1	3	1
6	G_1	CATG\$CTAGC	T_2	0	1	3	2
3	G_2	CTGCATG\$CT	A_0	0	1	3	3
1	T_0	AGCTGCATG\$	C_1	1	1	3	3
9	T_1	G\$CTAGCTGC	A_1	1	2	3	3
5	T_2	GCATG\$CTAG	C_2	2	2	3	3

C T G

Figure 3.8: Illustration of backward extension using the last-first (LF) property of the FM index. The rows represent sorted suffixes of the reference. The SA column holds the indices in the reference sequence for each suffix, the F column holds the first character of each suffix, while the L column holds the preceding character of each suffix. Subscripts give the occurrences of each character in each column. I.e. T_i is the ith occurrence of T in that column. Finding the string *CTG* is done by (1) starting with the range of all G's in the F column; (2) finding all T's in this range in the L column; (3) mapping by occurrence number those T's to the F column; (4) mapping the C'c in the current range in the L folder by occurrence number. The end result is in this case a single row which represents position 4 in the reference sequences.
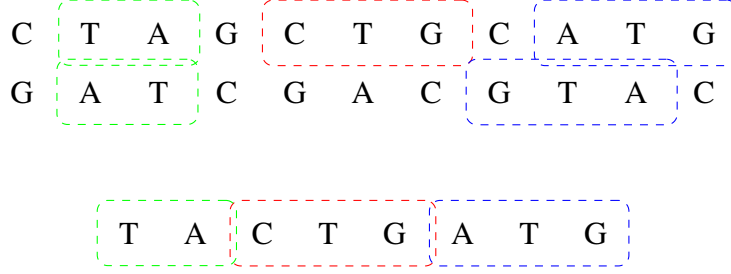
Figure 3.9: SMEMs found between query Q and both strands of reference R. Note that the MEM $(q, r, l) = (2, 0, 2)$ (CT) is not a SMEM, since it is contained in the SMEM $(q, r, l) = (2, 4, 3)$

threshold. Splitting SMEMs like this increases accuracy, since it increases the number of seeds, but can negatively affect performance. In order to find SMEMS, an adaption of the FM index is used, the FMD index, where $FMD(R) = FM(R * \bar{R})$.

Prioritizing and Extending

The seeds found from the index are next used as seeds for dynamic programming based alignment. Q is then aligned against the left and right side of $(r, r + l)$ for each seed (q, r, l). Since this is a computationally expensive step, further limiting the set of seeds is advantageous. BWA-mem does this by *chaining* the seeds. This is done by grouping approximately collinear, nearby seeds into chains, and removing small chains that overlap with larger chains. Approximately collinear means that $|(q_1 - q_2) - (r_1 - r_2)| < w$ for some set threshold w.

3.6.2 Graph Mapping

Mapping to a graph based reference is similar to the linear case, except that instead of finding a linear interval, the goal is to find a graph interval i_r such that $D(\ell(G_R(i_r), Q))$ is minimized. It is however deceptively more complicated. Firstly, because the number of subsequences in a sequence graph grows exponentially with the complexity of the graph, and secondly since chaining subsequence-matches is more complicated due to the possible existence of multiple paths between two matches. *vg* [9] has been at the forefront of mapping to a graph reference, showing that it can lead to better mapping accuracy than BWA-mem. Below is a brief description of the methodology used by *vg*.

vg

vg uses much the same methodology as BWA-mem to align reads. It uses the GCSA2-index to find SMEMs, uses chaining to filter the SMEMs, and uses a graph adaption of Smith-Waterman to extend the seeds. *vg* is able to align reads to more complicated graph structures than the simple directed sequence graphs considered in this thesis. For simplicity, the descriptions below will be contained to simple graphs, which entails that the GCSA index described is GCSA1[34] which is only able to index directed sequence graphs.

GCSA

The *Generalized compressed suffix array* (GCSA)-index [34, 35] is a generalization of the FM-index, where arbitrary-length sequences can be looked up in a sequence graph. Originally constructed to work on acyclic sequence graphs, version two of GCSA extends the functionality to general variation graphs. For simplicity we will here focus on acyclic graphs, i.e. version 1 of GCSA.

GCSA uses the same concept of LF mapping as the FM-index does. Here the F column holds the label of each node in the graph, sorted by the suffix starting from that node. The L column holds the labels of the corresponding node's predecessor nodes. The problem with this setup is that there are many suffixes starting from each node, depending on which path is taken in the graph. To resolve this problem, the graph needs to be expanded so that for each node, all suffixes starting from that node shares a prefix that is not found from any other node in the graph (see Figure 3.10).

For complicated regions in the graph, this expansion procedure gets too costly, and the GCSA index therefore needs to prune edges in such areas in order to be able to index it. This means that not all possible sequences in the graph gets indexed. Even after pruning, this procedure is costly and makes the GCSA index significantly slower to create, and more memory consuming than the FM-index.

Since the GCSA index provides the same functionality as an FM-index, it can be used to find SMEMs in much the same manner. These SMEMs are in *vg* used as seeds.

GCSA version 2 solves the problem in a slightly different way that allows for indexing general variation graphs. This is based on succinct de-Bruijn graph [2] structure which sets a limit to the length of the unique prefixes by allowing for false positive edges to occur in the index. This means that results from an index lookup need to be validated by traversing the graph.

Invalid

G – C – A – C – G
          T

Valid

G – C – A – C – G
      C – T

	Invalid				Valid	
SA		BWT		SA		BWT
	$	G_0			$	G_0
2	A_0	C_0		2	A_0	C_0
1	$C_{0,1}$	G_1		1	C_0	G_1
4	C_2	A_0, T_0		5	C_1	A_0, T_0
6	G_0	C_1		3	C_2	G_1
0	G_1	$		7	G_0	C_1
3	T_0	C_2		0	$G_{1,2}$	$
				4	T_0	C_2

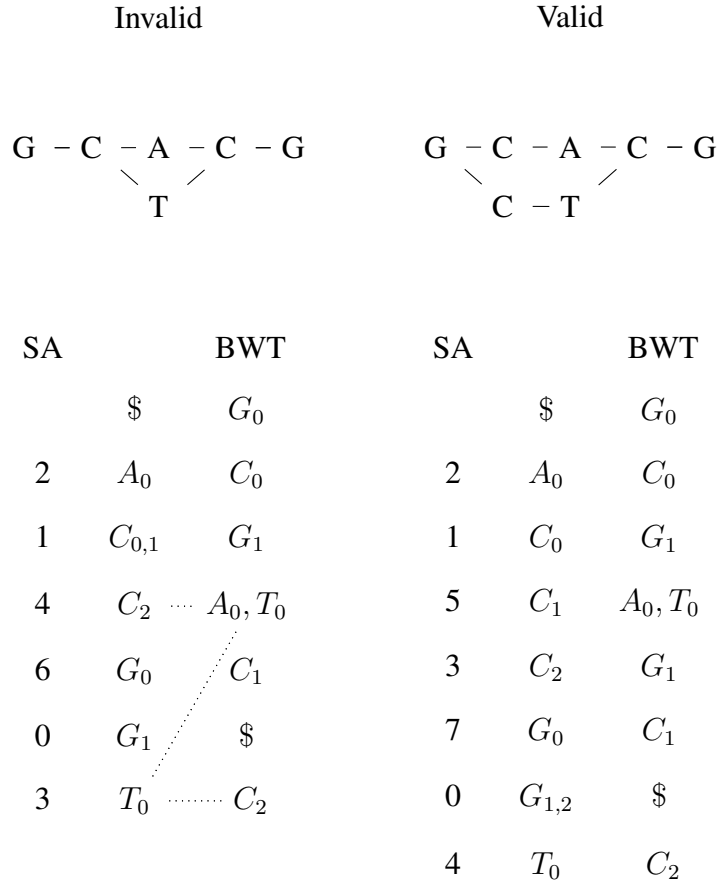Figure 3.10: Valid and invalid GCSA index. In the first graph, the two edges from C to A and T makes a direct GCSA index impossible due the other C with an edge to a G. Both T and G needs to point to the same row in the index. In the valid example, the C node have been duplicated. The bifurcation now happens at an earlier point, and the GC now present is unique so that the the LF mapping is valid.

Filtering and Extending

vg employs a similar method as BWA-mem for filtering out seeds: chaining the seeds and filtering out small chains that overlap bigger chains. The chaining procedure is however more complicated on a graph, and involves clustering the seeds by position in addition to a Markov Model based method*(cite)* to find approximately collinear seeds.

3.7   Peak Calling

Peak calling is an important step in the CHiP-seq experiment pipeline. The sequencing in a ChIP-seq experiments yields reads that tend to come from DNA-fragments with the transcription factor bound to it. After mapping these reads to the reference genome, the result is a set of genomic intervals which should be in the vicinity of the binding site. Peak calling is the process of using these intervals to predict regions containing the actual binding sites.

Calling peaks faces two main problems, noise and bias. Noise can come from either of the previous steps of the experiment: isolating DNA fragments with the TF bound to it, sequencing the fragments, or mapping the resulting reads to the reference genome. The bias can be either biological, in that certain regions of the genome are over/under-represented in the sample independently of the TF, or from the mapping, in that certain regions of the reference genome are more or less likely to be mapped to. In addition to this, the sequenced reads do not always contain the actual binding site, but rather can come from the beginning of a fragment containing a binding site.

In light of this, peak callers seek to recognize the pattern expected from a true binding site: that the mapped intervals should generally be close to and point towards the true binding site. Several algorithms exists for this [14, 39], of which MACS2 is one of the most widely used. As it forms the basis for Graph Peak Caller (paper 2), the algorithm is described in some detail in Figure 3.11.

```
Reads:          ACGTTCGTAT, TACAGCTCGA, TCGTAGCTAC, GTTCGTATAT
                GCTCGAGTAG , CTACTCGAGC, TATCTATCAA, TCAAACTACA


                ACGTTCGTATATCGTAGCTACTCGAGCTGTAGTTTGATAGATAT
Mapped:         [-------->.................................
                ......................<--------].............
                ...........[-------->.........................
                ..[-------->.................................
                .................<--------]..................
                ................[-------->..................
                ..................................<--------].
                ...........................<--------]......

            (0, 10, +), (21, 31, -), (7, 17, +), (2, 12, +)
            (17, 27, -), (17, 27, +), (33, 43, -), (27, 37, -)

Extended:       [--------+++++++++++++]....................
                .......[+++++++++++++--------].............
                ...........[--------+++++++++++++].........
                ..[--------+++++++++++++]..................
                ...[+++++++++++++--------].................
                .................[--------+++++++++++++]...
                ..................[+++++++++++++--------].
                ............[+++++++++++++--------].......

                (0, 24), (7, 31), (7, 31), (2, 26)
                (3, 27), (17, 41), (19, 43), (13, 37)

Pileup:         ...................-----.....................
                ................---------...................
                ............--------------..................
                .........-------------------...............
                ......------------------------.........
                ..----------------------------------......
                .------------------------------------...
                -------------------------------------.

Peak:           ...................-----...................
```

Figure 3.11: Illustration of the main MACS2 algorithm. After reads are mapped
to the reference genome, all mapped intervals are extended to match the estimated
fragment length. A pileup is then created based on the coverage of each position,
and positions with a high enough coverage are marked as peaks.

Chapter 4

Summary of Papers

4.0.1 Paper I: Coordinates and Intervals in Graph Based Reference Genomes

In paper I, we discussed the implications of using a graphical reference structure for representing and comparing genomic locations and intervals. The coordinate systems discussed are ones in which the graph is partitioned into disjoint paths where each path gets its own linear coordinate system. The disussion is about how the graph should be partitioned, and how the different paths should be named in order to provide an intuitve coordinate system that is robust against changes in the graph topology. The article further discuss how intervals should be represented on such graphs. Since there can be multiple paths betweeen two nodes in a graph, this is not as simple as on linear coordinate systems where intervals can be represented by a start and end position. The article describes two interpretations of intervals on grpahs, single-path intervals and multi-path intervals, and how they should be represented. A single-paths intervals is simply a path in the graph, while a multi-path interval are a subset of all paths between two nodes in the graph.

These consepts are examplified through an analysis of genes annotated on GRCh38 with alternative loci.

4.0.2 Paper II: Graph Peak Caller: Calling ChIP-seq peaks on graph-based reference genomes

Paper II developed a new method for calling ChIP-seq peaks on graph based reference genomes. Here we generalized the methodology used by MACS2 in order to utilize the increased accuracy of reads mapped to a graph based reference genome.

The new method is evaluated by comparing the motif enrichment in peaks called by Graph Peak Caller versus peaks called by MACS2 from the same reads. Using CHiP-seq data from *Arabidopsis thaliana* we show that the motif-enrichment is significantly higher for Graph Peak Caller peaks. In addition we show that the motif-enrichment is also higher a set of human and *Drosophila melanogaster* experiments.

4.0.3 Paper III: Assessing graph-based read mappers against a novel baseline approach highlights strengths and weaknesses of the current generation of methods

Paper III compares the performance of current graph based read mappers to a linear read mapper tuned for performance and not speed. This shows that the performance gains reported by graph based read mappers can be obtainable by increasing the searchspace of linear mapping. It also introduces a two-step approach to graph-mapping. Here, reads mapped to the graph is used to deduce a most likely path through the graph. This path is in turned into a linear reference which is mapped to using a linear reference. We show that this approach performs well on reads covering variants while avoiding the loss of accuracy due to mismappings.

4.0.4 Paper IV: Beware the Jaccard: the choice of metric is important and non-trivial in genomic colocalisation analysis

Paper IV investigates the properties different metrics used to measure genomic co-localization of binary tracks. We show that the different measures are affected differently by track size, and that the underlying reason for differing track sizes can be used to inform which similarity measure to use.

Chapter 5

Discussion

This thesis comprises three projects that investigates different facets of graph based reference genomes. In this work, and in the work of others, some problems have become clear which relate to the potentially large number of sequences in the language of a sequence graph. Foremost of these are the computational complexity of dealing with this large set of potential sequences, and the large number of sequences that does not exist in any observed individual.

5.0.5 Computational Complexity

The work in paper II and III involved using mapping reads using *vg*. Although the method used there is in principle very similar to what is used by BWA-mem, the memory requirements and running times are significantly higher. Graph Peak Caller itself also suffers from requiring more memory and having longer running times than MACS2. Although algorithmic and data structural advances might alleviate these problems, some complexities might not be resolvable without turning to further approximations, or changing the interpretation of sequence graphs. Indeed, Massimo et al showed that the indexing of sequence graphs is a hard problem with theoretical lower bound on complexity [7].

When the field of bioinformatics in recent years have been dominated by the sequencing capacity growing faster than Moore's law, it is worth to wonder how much accuracy gains is needed to justify the increase in computational complexity.

5.0.6 Invalid Sequences

Not only computational performance is affected by the large number of sequences in a graphs language. It can also lead to a large number of sequences that one would not expect to observe in nature. This is because all combinations of variants are represented in the language, but in nature variants can be highly positively of negatively correlated with each other. Such sequences makes graph mappers prone to mapping reads to sub sequences in the graph that matches the query string, but is unlikely to be the true origin of the read. This can lead to a loss of accuracy when mapping, but can also introduce biases in downstream analysis: Regions of the graph with many variants will be able to match many different query sequences and can thus become over represented when mapping reads to a graph. This potential bias is discussed in briefly in Paper II, where such over represented regions could be interpreted as peaks by the peak caller. The loss of accuracy in general can be seen in paper II in the relatively poor performance of graph mappers on reads not containing any variants. The two step approach introduced there can remedy such mapping effects, and it would be interesting in the future to use this approach also when calling ChIP-seq peaks, as this could remove the bias.

Recent work has introduced indexes for sequence graph that only index substrings that are present in one of the sequences which was used to create the graph [36]. This approach has the potential of both reducing the computational complexity, and improve the accuracy of the read mapping. It will be interesting to see how such approches perform in the benchmarks from Paper III, and also if they can improve the accuracy of downstream peak calling for ChIP-seq experiments.

Bibliography

[1] Bradley E Bernstein, John A Stamatoyannopoulos, Joseph F Costello, Bing Ren, Aleksandar Milosavljevic, Alexander Meissner, Manolis Kellis, Marco A Marra, Arthur L Beaudet, Joseph R Ecker, et al. The nih roadmap epigenomics mapping consortium. *Nature biotechnology*, 28(10):1045, 2010.

[2] Alexander Bowe, Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. Succinct de Bruijn Graphs. In Ben Raphael and Jijun Tang, editors, *Algorithms in Bioinformatics*, Lecture Notes in Computer Science, pages 225–235. Springer Berlin Heidelberg, 2012.

[3] Daniel Branton, David W Deamer, Andre Marziali, Hagan Bayley, Steven A Benner, Thomas Butler, Massimiliano Di Ventra, Slaven Garaj, Andrew Hibbs, Xiaohua Huang, Stevan B Jovanovich, Predrag S Krstic, Stuart Lindsay, Xinsheng Sean Ling, Carlos H Mastrangelo, Amit Meller, John S Oliver, Yuriy V Pershin, J Michael Ramsey, Robert Riehn, Gautam V Soni, Vincent Tabard Cossa, Meni Wanunu, Matthew Wiggin, and Jeffery A Schloss. The potential and challenges of nanopore sequencing. In *Nanoscience and Technology*, pages 261–268. Co-Published with Macmillan Publishers Ltd, UK, August 2009.

[4] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. 1994.

[5] The ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science*, 306(5696):636–640, October 2004.

[6] Giulio Disanto, Geir Kjetil Sandve, Vito AG Ricigliano, Julia Pakpoor, Antonio J Berlanga-Taylor, Adam E Handel, Jens Kuhle, Lars Holden, Corey T Watson, Gavin Giovannoni, et al. Dnase hypersensitive sites and association with multiple sclerosis. *Human molecular genetics*, 23(4):942–948, 2013.

[7] Massimo Equi, Roberto Grossi, and Veli Mäkinen. On the complexity of exact pattern matching in graphs: Binary strings and bounded degree. *arXiv preprint arXiv:1901.05264*, 2019.

[8] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 390–398, November 2000.

[9] Erik Garrison, Jouni Sirén, Adam M. Novak, Glenn Hickey, Jordan M. Eizenga, Eric T. Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F. Lin, Benedict Paten, and Richard Durbin. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36(9):875–879, September 2018.

[10] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, December 1982.

[11] J. Hein. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution*, 6(6):649–668, November 1989.

[12] Jotun Hein. TreeAlign. In Annette M. Griffin and Hugh G. Griffin, editors, *Computer Analysis of Sequence Data: Part II*, Methods in Molecular Biology, pages 349–364. Springer New York, Totowa, NJ, 1994.

[13] Fan Hsu, Hiram Clawson, Mark Diekhans, Robert M. Kuhn, W. James Kent, and David Haussler. The UCSC Known Genes. *Bioinformatics*, 22(9):1036–1046, 02 2006.

[14] Peter V. Kharchenko, Michael Y. Tolstorukov, and Peter J. Park. Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nature Biotechnology*, 26(12):1351–1359, December 2008.

[15] T. W. Lam, W. K. Sung, S. L. Tam, C. K. Wong, and S. M. Yiu. Compressed indexing and local alignment of DNA. *Bioinformatics*, 24(6):791–797, March 2008.

[16] Ben Langmead. Aligning Short Sequencing Reads with Bowtie. *Current Protocols in Bioinformatics*, 32(1):11.7.1–11.7.14, 2010.

[17] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357, 2012.

[18] Christopher Lee, Catherine Grasso, and Mark F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, March 2002.

[19] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[20] Heng Li. Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. *Bioinformatics*, 28(14):1838–1844, July 2012.

[21] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[22] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, July 2016.

[23] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009.

[24] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, 26(5):589–595, March 2010.

[25] Yongchao Liu and Bertil Schmidt. Long read alignment based on maximal exact match seeds. *Bioinformatics (Oxford, England)*, 28(18):i318–i324, September 2012.

[26] Udi Manber and Gene Myers. Suffix Arrays: A New Method for On-line String Searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[27] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[28] Benedict Paten, Adam M. Novak, Jordan M. Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome Research*, 27(5):665–676, May 2017.

[29] Aaron R. Quinlan and Ira M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, March 2010.

[30] Anthony Rhoads and Kin Fai Au. PacBio Sequencing and Its Applications. *Genomics, Proteomics & Bioinformatics*, 13(5):278–289, October 2015.

[31] Geir K. Sandve, Sveinung Gundersen, Halfdan Rydbeck, Ingrid K. Glad, Lars Holden, Marit Holden, Knut Liestøl, Trevor Clancy, Egil Ferkingstad, Morten Johansen, Vegard Nygaard, Eivind Tøstesen, Arnoldo Frigessi, and Eivind Hovig. The Genomic HyperBrowser: inferential genomics at the sequence level. *Genome Biology*, 11(12):R121, December 2010.

[32] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences of the United States of America*, 74(12):5463–5467, December 1977.

[33] Boris Simovski, Chakravarthi Kanduri, Sveinung Gundersen, Dmytro Titov, Diana Domanska, Christoph Bock, Lara Bossini-Castillo, Maria Chikina, Alexander Favorov, Ryan M. Layer, Andrey A. Mironov, Aaron R. Quinlan, Nathan C. Sheffield, Gosia Trynka, and Geir K. Sandve. Coloc-stats: a unified web interface to perform colocalization analysis of genomic features. *Nucleic Acids Research*, 46(W1):W186–W193, July 2018.

[34] J. Siren, N. Välimäki, and V. Mäkinen. Indexing Graphs for Path Queries with Applications in Genome Research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(2):375–388, March 2014.

[35] J. Sirén. Indexing Variation Graphs. In *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, Proceedings, pages 13–27. Society for Industrial and Applied Mathematics, January 2017.

[36] Jouni Sirén, Erik Garrison, Adam M. Novak, Benedict J. Paten, and Richard Durbin. Haplotype-aware graph indexes. In Laxmi Parida and Esko Ukkonen, editors, *18th International Workshop on Algorithms in Bioinformatics (WABI 2018)*, volume 113 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[37] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[38] Peter M Visscher, Matthew A Brown, Mark I McCarthy, and Jian Yang. Five years of gwas discovery. *The American Journal of Human Genetics*, 90(1):7–24, 2012.

[39] Yong Zhang, Tao Liu, Clifford A. Meyer, Jérôme Eeckhoute, David S. Johnson, Bradley E. Bernstein, Chad Nusbaum, Richard M. Myers, Myles Brown, Wei Li, and X. Shirley Liu. Model-based Analysis of ChIP-Seq (MACS). *Genome Biology*, 9(9):R137, September 2008.