1. Introduction

In this thesis four papers are presented that each relates to a different part of how reference genomes are used in bioinformatics. Read in the correct order, these can be seen to follow a common bioinformatics pipeline from start to end: Mapping sequenced reads to a reference genome (Paper III), using mapped reads to predict protein binding regions (Paper II), representing these binding regions as intervals on the reference (Paper I), and using these intervals in a colocalization analysis (Paper IV).

What separates this example pipeline from a normal bioinformatics pipeline is that the first three steps are performed using a graph based reference genome as opposed to a linear reference. The implications of using a graph based reference is the main focus of this thesis.

In order to give context to these papers and the concept of graph based reference structures, some background is provided in Chapter 2. Section 2.1 introduces important concepts from molecular biology, the subject matter which the methodologies used in this thesis are meant to elucidate. DNA-sequencing, the measuring process providing the raw data (sequenced reads) for the analyses in this thesis, is briefly explained in Section 2.2. Section 2.3 outlines how reference genomes are used to help make sense of this raw data.

Section 2.4 provides the mathematical framework used in the last, and most technical, part of the background which gives brief explanations of the relevant algorithms used to make inferences, using reference genomes, based on sequenced reads. The related concepts of sequence alignment (Section 2.5) and mapping (Section 2.6) is followed by peak calling (Section 2.7).

In total this background is meant to elucidate the role of reference genomes in bioinformatics, and provide a background for what a change to the use of graph based reference genomes entails.

Finally, Chapter 3 provides a brief summary of each paper provided, before the final discussion and conclusion of the thesis (Chapter 4).

2. Background

2.1 Molecular Biology

This section is meant as a very short introduction to the aspects of molecular biology most relevant for this thesis. For a more detailed explanations of the concepts, Part II of *Molecular Biology of the Cell* (**?**) is recommended.

2.1.1 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.

2.1.2 Replication and Mutation

In addition to serving as a 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 so that each of the two resulting cells have a copy of the DNA from the original cell. Furthermore, in sexual reproduction, germ line cells transfer 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 differences between cells within an organism, or differences 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.

2.1.3 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. In recent years, much attention has been devoted to the mechanisms determining gene expression levels. Two high-level consortia, ENCODE (**?**) and Roadmap Epigenomics (**?**), 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 in 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.

2.2 Sequencing

The fact that the sequence of nucleotides in the DNA has direct links to biological function has made determining such sequences as accurately and efficiently 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 (**?**), developed in 1976. The throughput from Sanger sequencing is pretty low since it requires measuring the weight of many nucleotides. In the early 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.

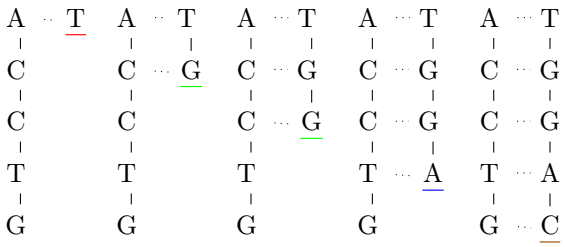


Figure 2.1: Sequencing of one DNA-chain in Illumina sequencing. For each step, an adapted nucleotide is added with a phosphorescent color, a picture is taken - capturing the color, and the adaption is reversed allowing for the next nucleotide to be bound. Each nucleotide (A, C, G, T) has a different color, so the sequence of colors in the images taken can be used to determine the sequence of nucleotides.

2.2.1 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 specific 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 further 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 2.1).

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 2.3, and the mapping process is covered in Section 2.6.

The main source of error from 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. For a more thorough introduction to the methods and applications of NGS, see (**?**).

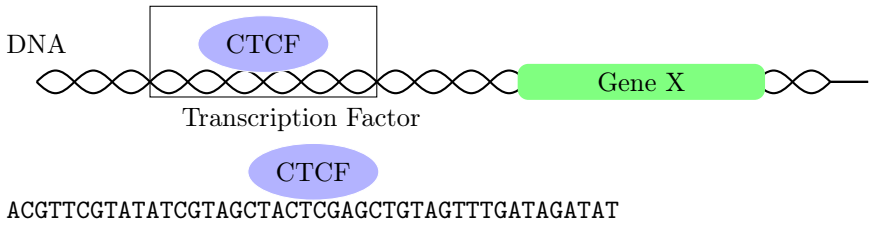
2.2.2 ChIP-seq

Chromatin Immunopresipitation followed by sequencing (ChIP-seq) is a sequencing experiment that does not aim to determine the genetic variation of the sample, but rather to give insight into 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 2.2). The process of analyzing these reads is covered in Section 2.7.

2.2.3 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 (**?**), passing DNA-molecules through a molecule which emits a different electric current for each type of nucleotide, and PacBio (**?**), 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 2.2: 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.

2.3 Reference Genomes

When it can be assumed that the genomic DNA-sequences from individuals of the same population are 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 2.6), 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.

2.3.1 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 (**?**). 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 (**?**) 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 region, this can give an indication that the gene closest to the binding region is relevant for the disease (see Figure 2.3).

Such analyses can be conducted on a genomic scale with the use of software like The Genomic Hyperbrowser (**?**) and others (**??**). This enables answering questions like whether variants associated with *Multiple Sclerosis* are overrepresented in accessible chromatin regions of immune cells (**?**). Paper IV in this thesis discusses how the choice of similarity measures between sets of genomic

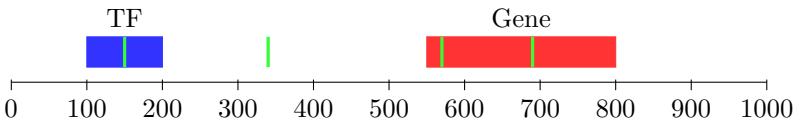


Figure 2.3: Figure showing the locations of several genomic elements on the same coordinate system. Locations of genes (red), transcription factor binding region (blue), and disease associated variants (green) from different sequencing experiments can all be compared and analyzed together. In this case two of the variants are located inside a known gene, and one is located in a predicted transcription factor binding site.

intervals can affect the outcome of such genome wide colocalization analyses.

2.3.2 Graph Based References

Both mapping of reads and any subsequent analysis of the predicted genomic features are affected by the reference genome used. The more the sample genome deviates 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 represent different variants is one way to achieve this (**?**). The first three papers of this thesis discuss various facets of a change from linear reference sequences to reference sequence graphs; Paper I discusses 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 2.5 and 2.6.

2.4 Mathematical Framework

The following sections introduce some notation needed for the next chapters. This notation will mainly follow a simplification of the notation used in (**?**).

2.4.1 Sequences

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^n_{DNA}. 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.

An *interval* on a sequenence is a tuple of start and end position (s, e). For an interval $i_S = (s, e)$, the notation $S[i_S]$ will mean the subsequence between the start and end coordinate $S[s : e]$.

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.

2.4.2 Graphs

A graph is a tuple $G = (V, E)$ of *vertices* (also referred to as *nodes*) 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. A *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 $\mathcal{P}(v_s, v_e)$. The *predecessors* of a node v are all nodes with an edge going to v, i.e: $\mathcal{T}^{-1}(v) = \{w \in V \mid (w, v) \in E\}$.

2.4.3 Sequence Graphs

We define a *sequence graph* $SG = (V, E, \ell)$ over an alphabet Γ as a graph (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 start and end nodes that are clearly defined, 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:

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

A *graph interval* is a path in the graph, and for a graph interval i_G, the notation $SG[i_G]$ will mean the label of that path, i.e. $\ell(i_G)$.

2.5 Alignment

2.5.1 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* (**?**), 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 $D(S, T)$, here meaning the Levenshtein distance unless specifically mentioned. The concepts discussed are however generalizable to other (weighted) edit distances and similarity measures 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 2.4).

2.5.2 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 $S[:M-1], T[:N-1]$ with a match or substitution at the end. Letting $d_{i,j} = D(S[:i], T[:j])$, we can write this as the recurrence relation

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

where

$$m_{k,l} = \begin{cases} 1 \text{ if } S[i] \neq T[j] \\ 0 \text{ otherwise.} \end{cases}$$

The initial conditions are given by $d_{k,0} = k$ and $d_{0,l} = l$ since the edit distance to an empty string is just the sequence length. The Needleman-Wunch algorithm (**?**) uses dynamic programming to calculate the matrix of edit distances d_{kl}, where d_{MN} is the edit distance between $S[:M]$ and $T[:N]$. 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 corresponds to a column in the alignment block. Figure 2.4 details the Needleman Wunch algorithm.

This algorithm can be adapted 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 (**?**), or one can use positive scores for matches to be able to find substrings that align well to each other (**?**).

An adaption relevant for this thesis, referred to as the *mapping* problem, is to find the substring 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 2.5 shows an example of

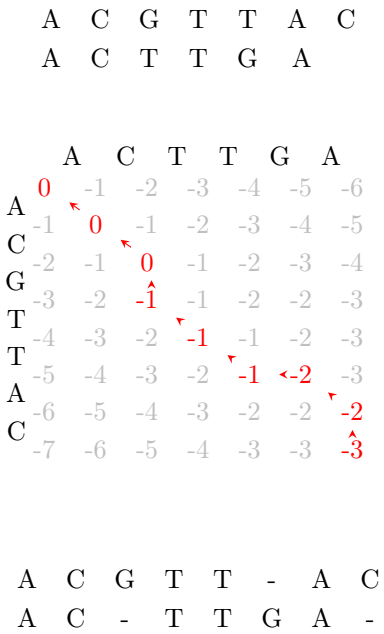


Figure 2.4: Figure showing the alignment of two sequences using Needleman-Wunch. Each cell in the matrix corresponds to the edit distance between prefixes of S and T. The red path shows the backtracking, resulting in an alignment block where each diagonal arrow gives a symbol from both sequences, while horizontal or vertical arrows give an insertion or deletion.

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 an exact, but slow, way the problem of the next chapter, namely mapping a read Q to a reference genome R.

2.5.3 Sequence Graph Alignment

The framework used to align sequences extends naturally to acyclic sequence graphs (**??**). 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 which yields the lowest edit distance. Similarly, the alignment of two sequence graphs $G = (V, E, \ell_G), H = (W, F, \ell_H)$, is the alignment of a sequence $S \in \mathcal{L}(G)$ to a sequence $T \in \mathcal{L}(H)$ that yields the lowest edit distance.

For each pair of nodes, $v_i \in V, w_i \in W$, we can define a distance between the two graphs up to the corresponding nodes as the lowest edit distance between two substrings ending at the given nodes,

$$d_{ij} = \min_{p_g \in \mathcal{P}(v_0, v_i), p_h \in \mathcal{P}(w_0, w_j)} D(G[p_g], H[p_h]).$$

We then get a recurrence relation similar to that of ordinary sequence alignment, except that all predecessor nodes of v_i and w_j have to be considered, not only $i-1$ and $j-1$ as in the linear case:

$$d_{ij} = \min \begin{cases} \min_{v_k \in \mathcal{T}^{-1}(v_i)} d_{kj} + 1 \\ \min_{w_l \in \mathcal{T}^{-1}(w_i)} d_{il} + 1 \\ \min_{v_k \in \mathcal{T}^{-1}(v_i), w_l \in \mathcal{T}^{-1}(w_j)} d_{kj} + m_{kl} \end{cases}$$

$$m_{kl} = \begin{cases} 1 \text{ if } \ell(v_k) \neq \ell(w_l) \\ 0 \text{ otherwise.} \end{cases}$$

If the graph is acyclic, then all the d_{ij} can be calculated using dynamic programming, without incurring any infinite loops.

The 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

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

This means that all sequences recognized by the alignment graph have the property that the sum of the distances 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

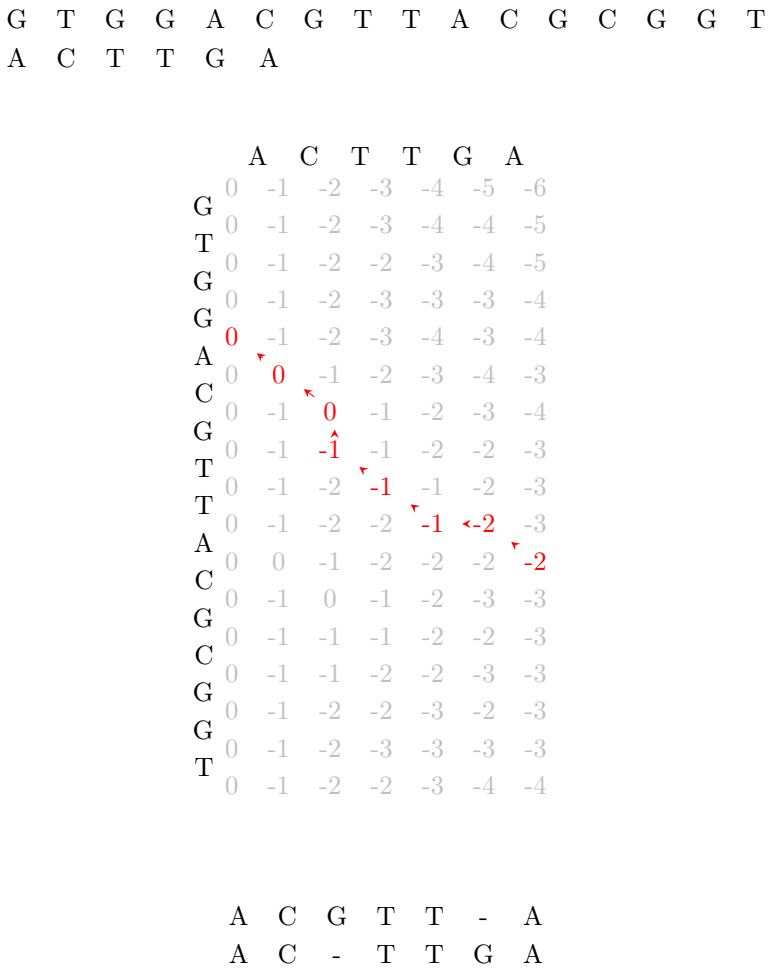


Figure 2.5: Figure showing the alignment of a query sequence Q to a reference sequence R, Each cell in the matrix corresponds to the edit distance between a prefix of S and a substring of R. The red path shows the backtracking, starting at the lowest value of the last column, resulting in an alignment block where each diagonal arrow gives a symbol from both sequences, while horizontal or vertical arrows give an insertion or deletion.

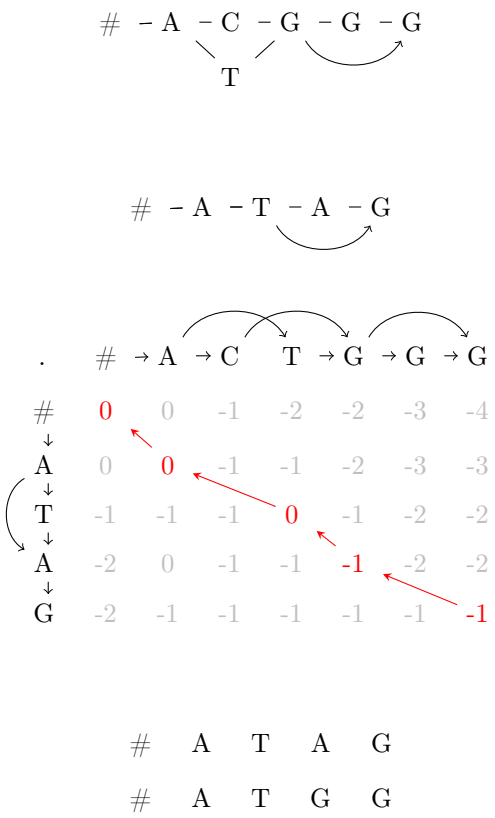


Figure 2.6: Figure showing the alignment of two sequence graphs. Marked in red is the path taken during backtracking. The result is the best alignment of a sequence from the language of each sequence graph. This alignment can again be represented as a sequence graph

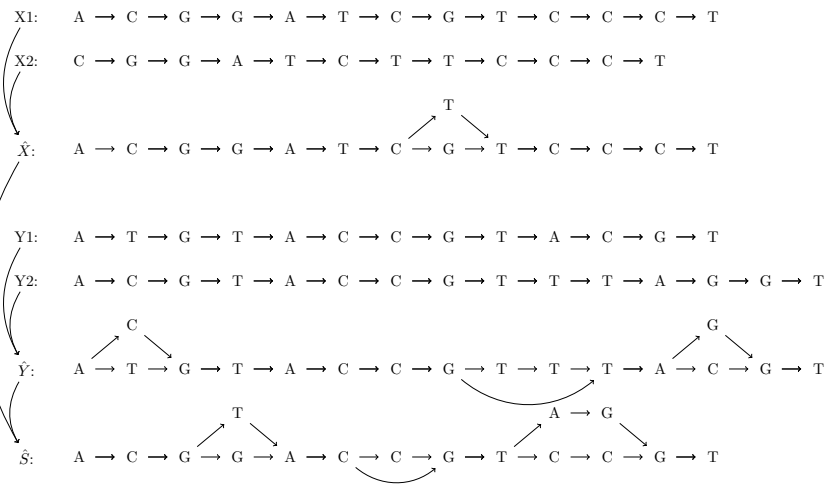


Figure 2.7: Iterative sequence graph alignments of four sequences evolved in two generations from $S = ACGTACGTACGT$. The sequences are represented as linear sequence graphs (X1, X2, Y1, Y2) and pairwise alignment is performed on the closest pairs yielding two sequence graphs $(\hat{X}, \hat{Y})$. These 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

finding the ancestors of two pairs of sequences that fit best together (**??**). This is illustrated in Figure 2.7.

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 2.6.2.

2.6 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 30-300 which need to be located in the human reference genome of length 3 billions. This would necessitate the calculation of 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 (**?**), which uses a common approach called the *seed-and-extend* paradigm. After this we will look at the methods used for graph alignment with focus on *vg* (**?**), which is also central in our work.

2.6.1 Linear Mapping

As in Section 2.5.2, we define the mapping problem as finding the interval i_R for wich the edit distance $D(R[i_R], Q)$ is smallest. We here introduce some of the heuristics used to solve this problem, with focus on the *seed-and-extend* paradigm. The seed-and-extend paradigm involves using an index to find substrings of the query sequence Q which has an exact match in the reference sequence R (called seeds) and using dynamic programming based methods to align the query sequence to a region around the seeds in the reference sequence. We let $EM(Q, R)$ be the set of all exact matches, represented by the tuples

$$EM(Q, R) = \{(q, r, l) \mid Q[q : q + l] = R[r : r + l]\}.$$

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.

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 (**?**) and variable-length indexes (**??**). 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) (**?**), described below.

FM-index

The FM-index uses a succinct representation of the suffix array (**?**) and Burrows-Wheeler transform (**?**) of the sequence in order to find exact matches of a query string S in $O(|S|)$ time (see Figure 2.8). 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 $O(|R|^{0.628} \times |Q|)$ time (**??**). Some algorithms also use the FM-index to find exact matches for permutations of the query sequence (**??**), 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 programming to intervals surrounding the seed matches on the reference sequence (**?**). One way of finding seeds is to find Maximal Exact Matches (MEM) (**?**). 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) (**?**). A SMEM is a MEM where no extension of the query interval has an Exact Match anywhere on the reference 2.9.

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

SMEMs are natural to use as seeds as they cover, for each substring 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 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 $FMD(R) = FM(R * \bar{R})$.

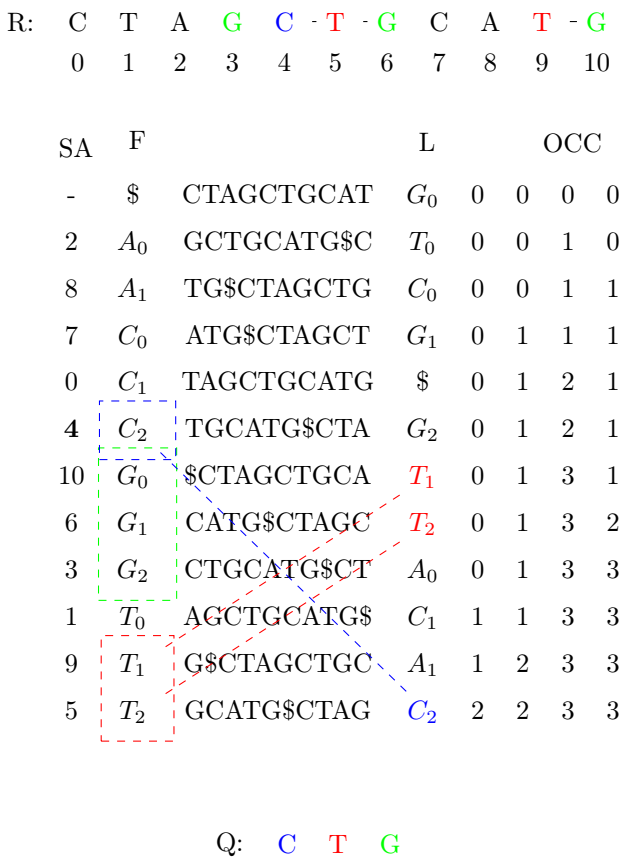


Figure 2.8: Illustration of backward extension using the last-first (LF) property of the FM index. The rows represent sorted suffixes of the reference. The SA column holds the indices in the reference sequence for each suffix, the F column holds the first character of each suffix, while the L column 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's in the current range in the L column by occurrence number. The end result is in this case a single row which represents position 4 in the reference sequences. The OCC matrix makes it quick to find the range of occurances of a symbol in the L column.

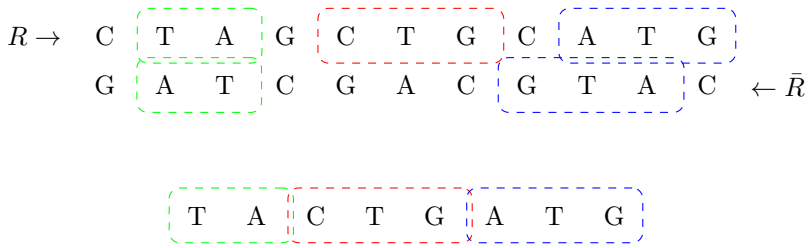


Figure 2.9: SMEMs found between query Q and both strands of reference R. Note that the MEM $(q, r, l) = (2, 0, 2)$ (CT) is not a SMEM, since the query interval is contained in the SMEM $(q, r, l) = (2, 4, 3)$

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.

2.6.2 Graph Mapping

Mapping to a graph based reference G_R 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 $D(G_R[i_r], Q)$ is minimized. It is however deceptively more complicated. Firstly, because the number of substring 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. The tool *vg* (**?**) has been at the forefront of mapping to a graph reference in company with *seven bridges* (**?**), 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 directed acyclic sequence graphs considered in this thesis.

GCSA

The *Generalized compressed suffix array* (GCSA)-index 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.

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 share a prefix that is not found from any other node in the graph (see Figure 2.10).

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 of general variation graphs. This is based on succinct de-Bruijn graph (**?**) 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.

Filtering and Extending

vg employs a similar method as BWA-mem for filtering out seeds: chaining the seeds and removing 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, to find approximately collinear seeds.

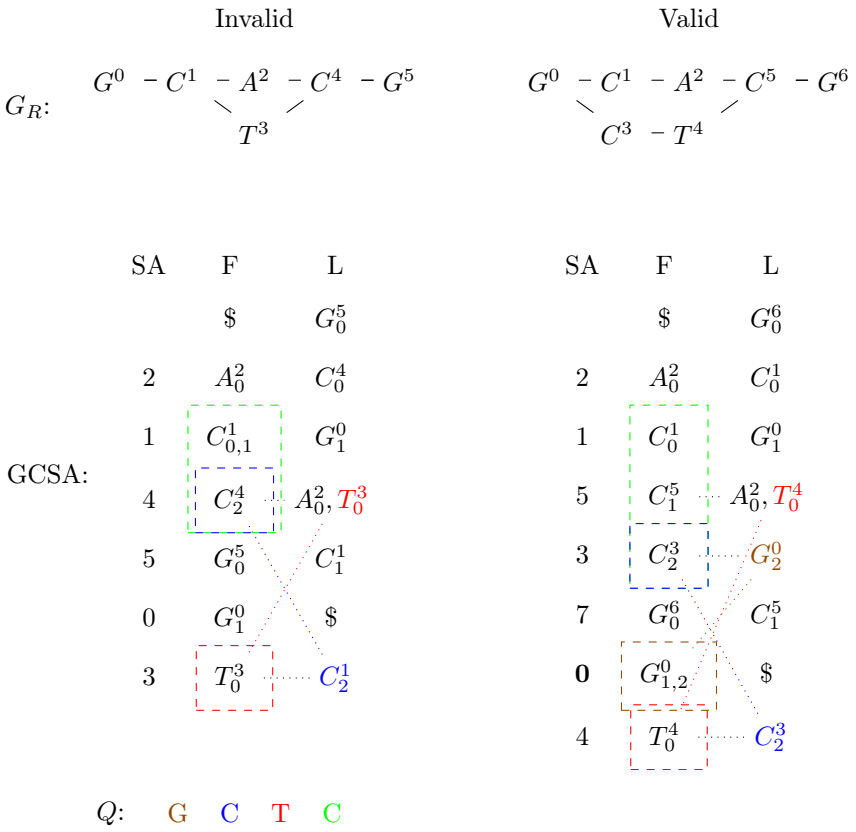


Figure 2.10: Example of a graph with an invalid GCSA index, and an expanded graph with the same language but a valid GCSA index. Superscripts are node IDs, while subscripts denote occurance in the index. In the first graph, the two edges from C^1 to A and T makes a direct GCSA index impossible due the other C^4 with an edge to a G. The suffix from C^4 (CG) comes between the two possible suffixes for C^1 ($CACG$, $CTCG$). This leads to the occurance of C^1 in the L and F columns being different, invalidating the LF-mapping. In the valid example, the C^1 node have been duplicated. The bifurcation now happens at an earlier point, and the GC prefix now common to all suffices from G^0 is unique so that the LF mapping is valid. This is examplified by the backward extension of the query sequence Q, which fails in the first index but is correct in the second index.

2.7 Peak Calling

Peak calling is an important step in the CHiP-seq experiment pipeline. The sequencing in a ChIP-seq experiment yields reads that tend to come from DNA-fragments with the transcription factor (TF) 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 (**??**), of which MACS2 is one of the most widely used. As it forms the basis for Graph Peak Caller (Paper II), the algorithm is described in some detail in Figure 2.11.

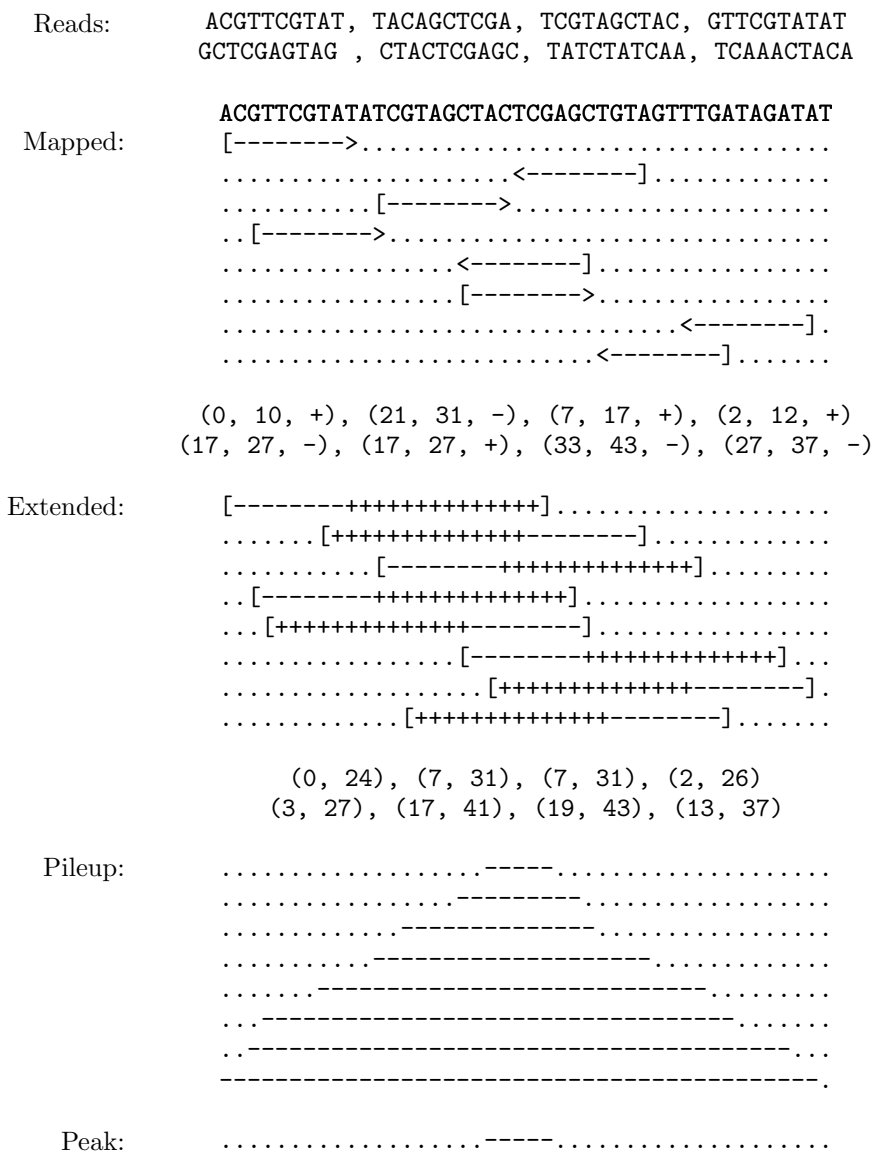


Figure 2.11: Illustration of the main MACS2 algorithm. After reads are mapped to the reference genome, all mapped intervals are extended to match the estimated fragment length. A pileup is then created based on the coverage of each position, and positions with high enough coverage are marked as peaks.

3. Summary of Papers

Paper I: Coordinates and Intervals in Graph Based Reference Genomes

In Paper I, we discussed the implications of using a graphical reference structure when 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 article discusses how the graph should be partitioned, and how the different paths should be named in order to provide an intuitive coordinate system that is robust against changes in the graph topology. The article further discusses 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 an end position. The article describes two interpretations of intervals on graphs, single-path intervals and multi-path intervals, and how they should be represented. A single-path interval is simply a path in the graph, while a multi-path interval is a subset of all paths between two nodes in the graph.

These consepts are exemplified through an analysis of genes annotated on the newest human reference genome (GRCh38) which includes alternative loci.

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 in a set of ChIP-seq experiments on human and *Drosophila melanogaster* samples.

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. Reads containing variants can be mapped more accurately with *vg*, but the higher rate of mismappings in general when mapping to a graph makes *vg* less accurate than the tuned linear mapper.

The article also introduces a two-step approach to graph-mapping. Here, reads mapped to the graph is used to deduce the most likely path through the graph. This path is turned into a linear reference which is mapped to using a linear mapper. We show that this approach performs well on reads covering variants, while avoiding the loss of accuracy due to mismappings.

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

Paper IV investigates the properties of different measures of similarity between binary tracks in the setting of genomic colocalization analysis where genomic features are represented as binary tracks. We show that commonly used similarity measures are affected differently by track size. Specifically, the Jaccard Index tends to yield higher values for larger tracks, while the Forbes index is not biased by track size, but has a higher variance for smaller tracks. We further show that the underlying reason for differing track sizes can be used to inform which similarity measure to use.

4. 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 do not exist in any observed sample.

4.1 Computational Complexity

The work in Papers II and III involved using mapping reads using *vg*. Although the method used there is in principle very similar to the method 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, **?** have shown that the indexing of sequence graphs is a hard problem with theoretical lower bound on complexity.

When the field of bioinformatics in recent years has 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.

4.2 Invalid Sequences

Not only computational performance is affected by the large number of sequences in a graph's 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 make graph mappers prone to mapping reads to substrings 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, and 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 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 III 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 avoid the bias.

Recent work has introduced indexes for sequence graphs that only index substrings that are present in one of the sequences which was used to create the graph (**?**). 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 approaches perform in the benchmarks from Paper III, and also if they can improve the accuracy of downstream peak calling for ChIP-seq experiments.

4.3 Conclusion

In this thesis, the role of reference genomes has been investigated from several angles with a focus on how a change to a graph based representation can affect commonly performed analyses. As discussed above, a graph based representation carries with it some inherent complexities. However, these complexities aside, we have shown that a graph based approach can increase accuracy: The two step graph mapper introduced in Paper III shows improved mapping accuracy, and Graph Peak Caller (Paper II) seems from the motif enrichment analysis to predict binding regions more accurately than MACS2. These results can provide motivation for overcoming the complexities, both computational and conceptual, of working with graph based reference structures in order to attain the improved accuracy with tools that are easy to understand and use, and without the high computational requirements.