Master Thesis

Handling ambiguity in read mapping applications

Sven-Léon Kuchenbecker

October 20, 2011

Supervisors
Prof. Dr. Knut Reinert
Prof. Dr. Martin Vingron

This thesis deals with the problem of post-processing read mapping data to resolve ambiguously mapped reads, i.e. regions within the multi-alignment produced by a read mapper that are covered by reads that actually originate from different regions. For that purpose, an algorithm work flow was designed, implemented and evaluated for different read mapping scenarios.

Contents

I

Introduction

This work discusses an approach to handle ambiguity in read-mapping applications. Ambiguity, in this scenario, describes the case where regions in the reference sequence have reads mapped against them, which are not all of common origin.

In this chapter, an introduction to DNA sequencing methods and the computational processing of the resulting data, namely sequence assembly and read mapping is given. Furthermore, the impact of read ambiguity during these processes is discussed for different scenarios and applications. The chapter then closes with previous work on the subject and a description of the goals of this thesis.

1.1. DNA Sequencing

DNA sequencing methods can be roughly divided into two categories – the automated Sanger method, sometimes also referred to as *first generation sequencing*, and the recently developed *next generation sequencing* methods, a term describing a group of recently emerged high throughput methods [1].

Automated Sanger sequencing has been the de facto standard for sequencing for almost 20 years and has been involved in most of the major accomplishments in the field of DNA sequencing, including the human genome project. The principle of the Sanger method [2] is to determine the sequence by interrupted DNA replication using primers, dNTPs and DNA polymerase similar to those used in PCR setups. To add a known primer binding site and for clonal amplification, the target fragment is inserted into a vector and incorporated into a bacterial cell. It is then clonally amplified, extracted and cut out of the vector. The resulting DNA

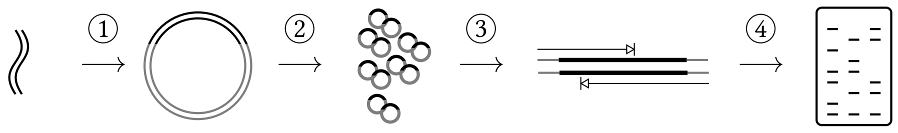


Figure 1.1.: A simplified scheme of the Sanger sequencing approach. The DNA strand of interest (template) is inserted into a vector ① and incorporated into a bacterial cell. Afterwards, the host cell is cloned ② and the insert of each vector is transcribed using primers sensitive to the vector and a mixture of regular dNTPs as well as transcription terminating ddNTPs with fluorescence markers ③. The sequence is recovered with high resolution gel electrophoresis using the segments lengths and the detected fluorescence ④.

molecules are then replicated in vitro, where in addition to the regular dNTPs a small fraction of dideoxynucleotides (ddNTPs) is added that, if used by the DNA polymerase to elongate the DNA strand currently created, terminate the elongation of the new strand. The ddNTPs of each base type are labeled with different fluorescence markers. The replication is then performed on the clonally created copies of the same fragment. Due to the mixture of dNTPs with ddNTPs the transcription of each copy is terminated at a different stage and fragments of different lengths are produced. The sequence of the original strand can then be recovered by analysing the length and terminating base of each fragment by gel electrophoresis and fluorescence analysis [3]. Sequences recovered from DNA molecules using DNA sequencing are called *reads*. The basic work flow of the Sanger sequencing method is illustrated in Figure 1.1.

Next generation sequencing methods also use the DNA replication mechanism with fluorescent terminal ddNTPs to recover the base types. However, in contrast to original Sanger sequencing, these methods allow to read multiple bases from the same DNA molecule while it is being elongated and can process many different template fragments at the same time. There are two major requirements for this to be possible: for one thing, the different template fragments needs to be spatially separated and immobilized, such that the subsequent fluorescent events for each base can be associated with the fragment they originate from. Secondly, the elongation termination needs to be reversible, such that the elongation can continue after the most recently added base type has been detected. Different approaches have been developed to meet both requirements, resulting in different sequencing platforms. As an example, in the *Illumina/Solexa* sequencer the template DNA molecules are immobilized using a glass plate with forward and reverse primers attached to it. The template fragments bind to those primers and are then amplified while spatially fixated using free neighboring primers, a process called *bridge amplification*. This way, each template builds up a cluster of spatially fixated copies on the glass plate. After this amplification step, the base by base replication is started by adding dye labeled nucleotides containing a termination group. After the replication of the first base is completed for all fragments, the free nucleotides are washed away and the signal is detected. Then, the termination groups and the dyes are cleaved from the nucleotides and the process is repeated until the elongation has finished.

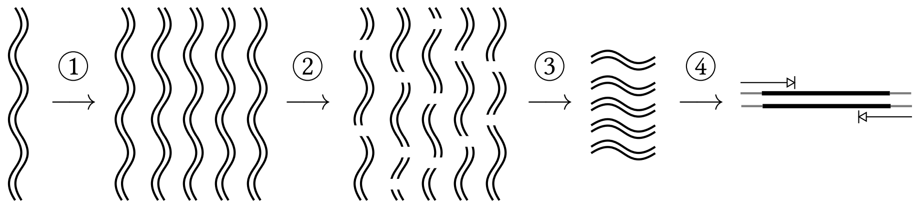


Figure 1.2.: An illustration of the shotgun sequencing steps. The original DNA strand is copied various times ① and then cleaved at random positions ②. Those fragments that have the right length for the chosen sequencing method are selected ③ and both ends of each fragment are sequenced ④.

Developments as the *Illumina/Solexa* sequencer and other next generation sequencing platforms have caused a significant decrease in costs for DNA sequencing. Before 2007, when the first of these platforms appeared, the costs for automated Sanger sequencing had dropped to about 500 USD per megabase. With next generation sequencing platforms these costs have dropped further and more rapidly to less than 0.5 USD per megabase until 2011 [4]. A disadvantage of next generation platforms however are the read lengths, which have been significantly smaller for the early platforms (approx. 30 base pairs) and still have not reached the read lengths of automated Sanger sequencing, which produces about 500-1000 base pairs per read. The read lengths for next generation platforms are however increasing rapidly, e.g. newer *Illumina/Solexa* platforms produce reads of length 150.

Nevertheless, independent of first or next generation sequencing methods being used, the maximum read length is a limitation in many applications. Available sequencing methods cannot be used to directly sequence entire genes (a human gene spans in average a region of 25-30,000 base pairs [5]) or even genomes (the human genome contains about 3.3 billion base pairs [5]). The most popular approach to deal with this limitation is the so called *shotgun sequencing* method, as illustrated in Figure 1.2. The original sequence is duplicated several times and each copy is cleaved into smaller pieces at random positions (e.g. using ultrasonic waves). From the random fragments those that have the right size for the chosen sequencing method are selected and sequenced. The result is a set of overlapping read sequences that can then be processed computationally with the goal to determine the relative position of the reads to each other or to a reference sequence. To support this process, additional anchor points between the reads are added by using a template size larger than the read length and sequencing it from both ends. That way, for each template two reads are produced, whose relative position to each other can be estimated by their length and the chosen template length. Such a pair of reads is called a *mate pair*. The type of read processing depends on the application and can be divided into *sequence assembly* and *read mapping*, which are described in more detail in the following sections.

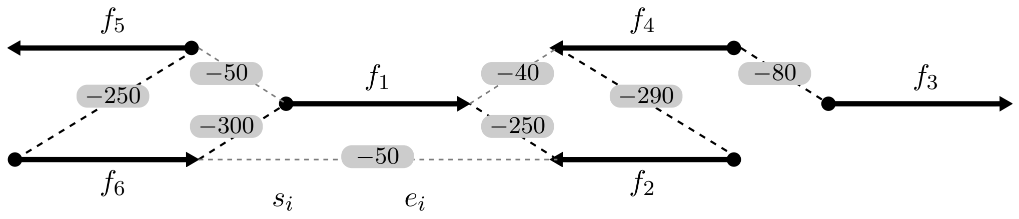


Figure 1.3.: An example overlap graph (modified from [6]). All reads have a length of 500bp. The start nodes s_i are denoted with ●—, the end nodes e_i are denoted with ⟶. The minimum spanning tree is highlighted in black.

1.2. Sequence assembly

If the sequence of an entirely unknown DNA molecule is of interest, a so called sequence assembly is performed. Given the set of reads $\mathcal{F} = \{f_1, \ldots, f_n\}$ resulting from the shotgun sequencing of the original sequence $A = a_1, a_2, \ldots, a_L$, the goal of a sequence assembly is to recover the original sequence A using the information available due to the overlaps of the reads. A standard sequence assembly approach can be divided into three stages: the overlap phase, the layout phase and the consensus phase [6].

1.2.1. The overlap phase

During the overlap phase the overlap between every pair of reads is computed. The result is stored in a special data structure, called an *overlap graph*. For a set of reads $\mathcal{F} = \{f_1, \cdots, f_n\}$ the overlap graph is defined as $G(V, E = E_1 \cup E_2, \omega)$, where

- $V = \bigcup_i \{s_i, e_i\}$ contains two vertices for each read $f_i \in \mathcal{F}$ representing its start and end.
- E_1 contains a directed edge $e_1^i = (s_i, e_i)$ for each f_i, where the edge is weighted with $\omega(e_1^i) = \text{length}(f_i)$.
- E_2 contains undirected edges $e_2^j = \{v_k, v_l\}, v_k, v_l \in V$, representing an overlap between two reads. The edges are weighted with $\omega(e_2^j) = -l_{k,l}$, where $l_{k,l}$ is the length of the overlap between reads f_k and f_l. v_1 (and v_2 resp.) is chosen depending on which part of the read (beginning or end) overlaps with the other read.

An example graph is shown in Figure 1.3. Note that the overlap graph does not have to be (and is mostly not) connected. Typically, the set of reads contains groups of overlapping reads, hence the assembly result will not be a continuous sequence but a set of subsequences, so called *contigs*.

1.2.2. The layout phase

In the layout phase the information stored in the overlap graph is used to compute a multi-alignment of the reads. This is mostly done under the assumption that the most likely layout is the most compact layout, or put differently, that the *shortest common superstring* is the correct consensus sequence for a given set of reads. Finding the shortest common superstring can be reformulated as a graph problem on the overlap graph, namely finding the *minimum spanning tree* for each connected component of the overlap graph. The minimum spanning tree of a graph $G(V, E, \omega)$ is defined as the tree $T(V, E' \subset E)$, such that $\sum_{e \in E'} \omega(e)$ is minimal. The minimum spanning tree for the example shown in Figure 1.3 is highlighted in black.

Finding the shortest common superstring (or the minimum spanning tree respectively) is known to be NP-hard [7]. Due to the complexity of the problem, mostly heuristic approaches are used, such as choosing the overlap edges in decreasing overlap length.

1.2.3. The consensus phase

In the consensus phase the final layout, i.e. a multi-alignment that takes into account the overlaps that are referenced by the realized overlap edges in the minimum spanning tree and the consensus sequence of that multi-alignment is computed.

1.3. Read mapping against a reference

Besides reconstructing the consensus sequence from the reads produced by the shotgun sequencing, another application is to map the reads against a reference sequence or a database of sequences. The *read mapping* problem can be formulated like this: Given the set of reads $\mathcal{F} = \{f_1, \ldots, f_n\}$ a reference sequence G and a distance threshold $k \in \mathbb{N}$, find all substrings g of G with distance at most k to any $f \in \mathcal{F}$, called matches [8].

Mapping a read against a reference is mostly performed in two stages: first, a filtering algorithm is applied that is supposed to reduce the target regions within the reference sequence significantly, such that the expensive search for approximate hits does not have to be performed on the entire reference sequence. The filter uses an index data structure such as the q-gram index (in the case of the QUASAR [9] or SWIFT [10] filters), returns candidate regions in the reference sequence and guarantees that there are no matches outside those regions. In a second step, the candidate regions are checked for matches.

1.3.1. Applications

As next generation platforms drastically reduced costs for DNA sequencing, a lot of new applications appeared. The focus was extended from sequencing only a few reference genomes for each species to actually analyse sequences of individuals and their variations. Most of these new applications are read mapping applications.

Resequencing

Large scale sequencing of individuals of the same species has been impossible within reasonable cost and time constraints with traditional sequencing methods. Now however, *whole genome resequencing*, i.e. variation analysis between individuals on a genomic level became affordable [11]. Instead of reassembling the reads again, the reads are mapped against the already assembled reference genome of another individual of the same species. A suitable distance threshold covers the *single nucleotide polymorphisms (SNPs)* between the individuals, which can then be detected and analysed. To further reduce costs, *exome sequencing* enables us to selectively capture only the exome, i.e. the protein coding part of the genome. While the human genome has an overall length of approximately 3 Gb (3 billion base pairs), only about 1% (~30 Mb) of the overall sequence is actually encoding for proteins [12].

Quantitative methods

For a long time, gene expression analysis (or more precisely transcriptome analysis) has been done by hybridizing reverse transcribed mRNA with cDNA probes on microarray chips. With next generation sequencing platforms it became more attractive to do this directly on the sequence level. *RNA-Seq* does exactly that - instead of hybridizing the DNA molecules with reference molecules they are sequenced and the reads are mapped against reference sequences. Quantification is then done based on the coverage of certain reference locations. RNA-Seq is a relatively new method and still under development, but it is expected to overcome some disadvantages of microarrays such as lack of comparability across experiments and background noise due to cross-hybridization [13].

Another method that has recently been migrated from microarray- to sequence-based evaluation is *Chromatin immunoprecipitation (ChIP)*, a method to investigate interactions between DNA molecules and proteins in the cell. It allows the selection of DNA fragments from a cell that are bound to a particular protein. Until recently the ChIP-DNA has been hybridized on microarrays for analysis (*ChIP-Chip*) – now also these analyses are performed by sequencing and mapping the fragments instead (*ChIP-Seq*) [14].

Metagenomics

Metagenomics is the study of the microbial DNA content of environmental probes. It enables us to study microbes that are not culturable, which actually applies to about 99% of all microorganisms [15]. While early approaches involved constructing BAC libraries from the environmental DNA samples, screening those for phylogenetic markers and eventually using Sanger sequencing to recover sequence fragments of the taxa of interest, nowadays the entire sample can be sequenced and the reads are assigned to taxa by mapping them against reference databases [16].

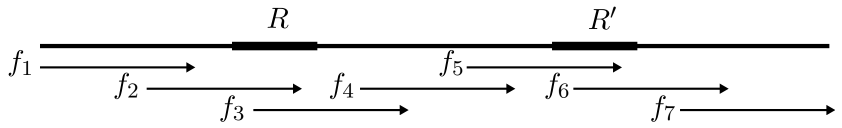


Figure 1.4.: A simplified example of an original sequence containing two repeats (R and R') and a read layout that will be falsely reconstructed (modified from [6]).

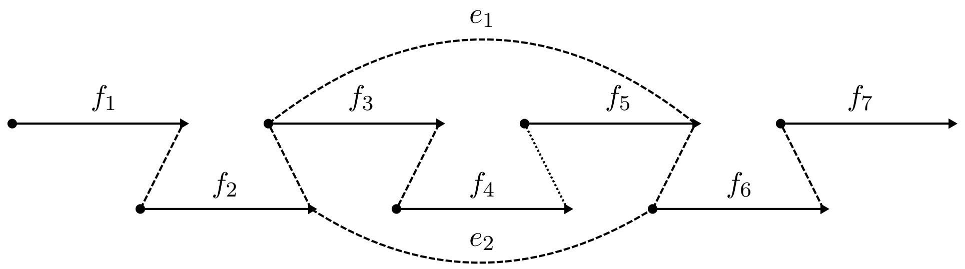


Figure 1.5.: The overlap graph produced from the reads shown in Figure 1.4. The graph contains two repeat induced false overlap edges, e_1 and e_2 (modified from [6]).

1.4. Read ambiguity

The term *read ambiguity* shall be defined to generally describe the situation when two or more reads that originate from different locations within the sequenced sample are highly similar, such that software computing a read layout will assign them the same or overlapping locations. Both in sequence assembly and in read mapping applications read ambiguity can be (and is in practice) a problem. In the case of sequence assembly, similar or identical reads originate from repeats within the original sequence and cause repeat occurrences to be *compressed* into one occurrence. In read mapping similar or identical reads will be mapped against the same location, even though they are not of common origin and thus falsify the result in most applications.

1.4.1. Effects in Sequence Assembly

The critical phase concerning repeats during sequence assembly is the overlap phase. Repeat regions within the original sequence cause false overlap edges in the overlap graph and thus falsify the information used for computing the read layout. Consider the example shown in Figure 1.4. The reads f_2, f_3, f_5 and f_6 cover the repeat regions R and R'. After computing all pairwise overlaps during the overlap phase, an overlap between f_2 and f_6 as well as an overlap between f_3 and f_5 will be detected. These overlaps are however induced by the repeats and not by the fact that the fragments originate from the same region on the original sequence. The resulting overlap graph is shown in Figure 1.5. While this particular situation is still relatively easy to detect and resolve, the problem becomes more complex when the repeat lengths significantly exceed the length of a read, i.e. there are reads that originate from repeat regions

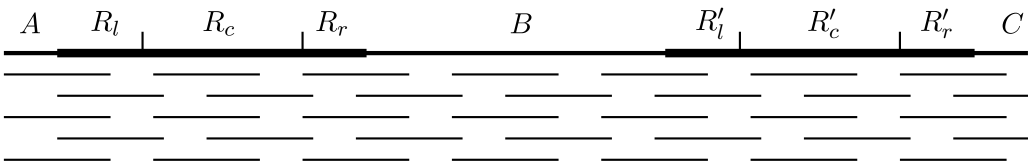


Figure 1.6.: The **original** sequence (up) contains two repeats, R and R', as well as three unique regions, A,B and C. Below the sequence the correct read layout is shown (modified from [6]).

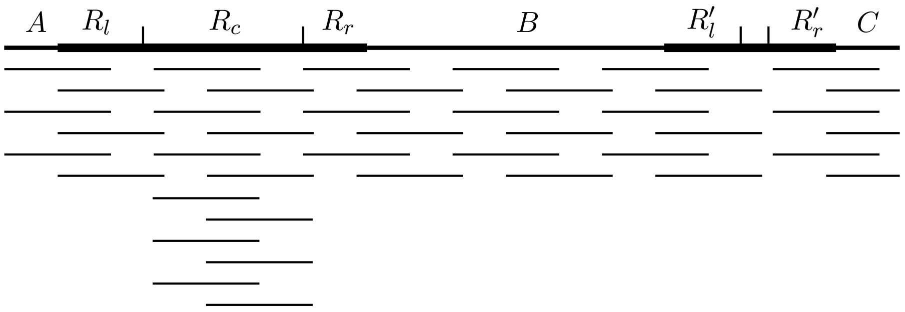


Figure 1.7.: The **reconstructed** sequence (up) no longer contains both repeats completely, the region R'_c without overlaps to any of the unique regions has been overcompressed. The reads have been assigned to the region R_c, which shows a higher coverage as a result (modified from [6]).

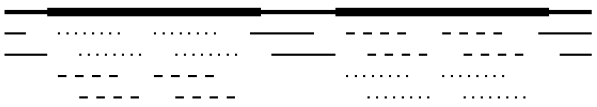
and do not overlap with any unique region within the original sequence. The additional edges within the overlap graph compose shorter paths to be used by the minimum spanning tree and hence shorten the consensus layout (and sequence).

The effect is illustrated in Figures 1.6 and 1.7. The reads that do not overlap with any of the unique regions A, B or C are compressed into one occurrence in the consensus sequence.

1.4.2. Effects in read mapping

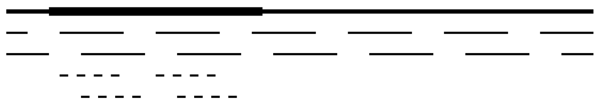
In read mapping, read ambiguity can be roughly divided into three categories. In any case, the layout proposed by the read mapper contains locations covered by at least two reads that do *not* originate from the same sequence or location.

1. **The reference sequence and the input strand both contain (the same) similar regions**

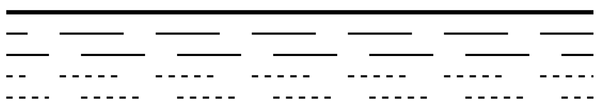
   When the reference contains similar regions, reads that match against one of them will match against all of them, if the distance threshold is not exceeded. The affected regions show unexpectedly high coverage and the reads will map against multiple locations (*multireads*).

2. **The reference sequence contains only some of the similar regions present in the input strand**



If the input sequence contained similar regions $\mathcal{R} = \{R_1, \dots, R_n\}$, but the reference contains only a subset $\mathcal{R}' \subset \mathcal{R}$, then the reads from those "hidden" subsequences will be mapped against the available regions. A possible reason for this situation is repeat compression during the assembly of the reference sequence (as described in Section 1.4.1). If $|R'| = 1$ all reads can be mapped uniquely, thus there are no multireads. The affected locations within the reference sequence show a higher coverage.

3. **Multiple input strands (mixed sample)**



If the input sequence contains multiple haplotypes, the reads can be seen as sequences originating from similar regions as well. They are however equally distributed among the reference sequence and induce neither multireads nor additional coverage.

Depending on the application the effects of the false mappings are different. Quantitative methods such as CHiP-Seq and RNA-Seq will suffer from false high quantifications [17], whereas in resequencing methods the consensus and hence the detected variations are falsified.

Also it should be noted that even though in some of the scenarios described above read ambiguity will result in a higher coverage for the affected regions, that does not necessarily imply that the difference in coverage can be used to identify such regions. In some applications that might be the case, in others however, in particular in quantitative methods, where reference regions are expected to show significant differences regarding their coverage, it is not.

1.5. Micro heterogeneity

If two reads f_1 and f_2 that map against the same location even though they do not originate from the same location on the input sequence, are perfectly identical, there is obviously no way to classify them according to their origin. Fortunately, perfect sequence conservation is something that rarely occurs in real life - hence differences between the reads are to be expected. An example of that *micro heterogeneity* is shown in Figure 1.8. The columns that contain multiple deviations of the same character induced by read ambiguity are denoted as *separating columns*, since the information contained in these columns can be used to classify the reads according to their origin. This information can be supported by other columns that contain deviations at the very same positions.

```
reference  ATGTACTAGCAGCTCGCACCCCTGGTTTAAGAGTTGTAATTGGGGCGGTAAC
    reads  ATGTACTAGCAGCTCGCACCCCTGGTTTAA
           ATGTACTAGCAGCTCGCACCCCTGGTTTAAGAGTTGTAATTGGGGCGGTAAC
           ATGGACTAGCAGCTCGCACCCCAGGTTAAAAGTGTTTTAATTGGGGCGCTAAC
           ATGTACTAGCAGCTCGCACCCCTGGTTTAAGAGTTGTAATTGGGGCGGTAAC
           ATGGACTAGCAGCTCGCACCCCAGGTTAAAAGTGTTTTAATTGGGGCGCTAAC
           ATGGACTAGCAGCTCGCACCCCAGGTTAAAAGTGTTTTAATTGGGGCGCTAAC
                            ACCCCTGGTTTAAGAGTTGTAATTGGGGCGGTAAC
```

Figure 1.8.: Correlating differences within the read sequence. Characters deviating from the column consensus are highlighted. This illustrative example contains rather many of those deviations, as opposed to the expected number of deviations in practical applications.

1.5.1. Previous work and contents of this thesis

A fundamental step of every algorithm classifying reads based on micro heterogeneity is to decide if a column in the consensus layout qualifies as a separating column, i.e. if its deviations are statistically significant or if the observed deviations simply occur by chance. *Kececioglu and Ju* [18] as well as *Tammi et al.* [19] developed criteria to quantify the significance of candidate columns. Apart from that, *Kececioglu and Ju* also proposed an entire algorithm work flow to perform the classification and integrate the results into the sequence assembly process.

Classification based on micro heterogeneity can be seen as the most general approach to handle read ambiguity, since it uses a very low level type of information and is application independent. There are however also other, application specific approaches, e.g. for quantitative methods. *Mortazavi et al.* [17] analyzed the effects of mapping uncertainty in RNA-Seq expression estimations and suggested to distribute multireads proportional to unique reads that map to the same regions to correct for the bias they introduce. Furthermore, *Li et al.* [20] developed a more sophisticated method using a Bayesian Network modeling the RNA-Seq sequencing process to estimate expression levels, taking into account also read ambiguity.

The goal of this thesis was to build a work flow to resolve read ambiguity in read mapping applications using micro heterogeneity. As an initial basis the work flow developed by *Kececioglu and Ju* to perform repeat resolution during sequence assembly was taken and adapted to be suitable as a post processing refinement step to improve the result of a read mapper. That adaption includes an alternate iteration through the data, namely the multi- alignment produced by the read mapper as opposed to the original approach, which alternates between analysing the current read layout and modifying the conflict graph. Furthermore, the significance test developed by *Tammi et al.* is integrated into the adapted pipeline, as it promises to be more suitable for shorter read lengths and read mapping applications. Finally, some application dependent post processing methods were added, integrating the local classifications of read occurrences within the multi-alignment depending on the goal of the separation - either the correction of the multi-alignment, i.e. the removal of falsely mapped reads or the creation of larger classes of read occurrences that might make the recovery of hidden origins possible.

All of these methods will be discussed in Chapter II.

The methods discussed in this thesis were also implemented and evaluated. The performance of the separation methods in different scenarios was compared for different criteria, such as the sensitivity regarding read occurrences that require separation, the precision in detecting separating columns, the quality, i.e. homogeneity of the produced classes etc. The results of those evaluations are described in Chapter III.

II

Algorithms

In this chapter an algorithm work flow for ambiguity resolution on a multi-alignment produced by a read mapper based on micro heterogeneity is described. It utilizes methods for handling read ambiguity during the read assembly process proposed by *Kececioglu and Ju* [18], which were adapted and extended to be applied as a post processing step after read mapping. Also, an alternative method to evaluate the significance of deviations in read sequences for ambiguity handling proposed by *Tammi et al.* [19] is described as a substitute for the original method.

2.1. Preliminaries

Given a reference sequence s_{ref} of length g and a set of N read sequences $\mathcal{F} = \{f_1, \ldots, f_N\}$, the set of all read occurrences detected by a read mapper given s_{ref} and $\mathcal{R}$ as input is denoted as $\mathcal{L}$. Each successfully mapped read is represented by one or more elements $l = (\ell^s, \ell^e, \ell^r) \in \mathcal{L}$, where ℓ^s is the start position within the multi-alignment, ℓ^e the end position and l^r the index of the read sequence occurring at that location. In other words, there exists an alignment of the sequence $r_{\ell^r} \in \mathcal{R}$ that starts at column ℓ^s and ends at column ℓ^e of the multi-alignment and fulfills the constraints specified by the user when calling the read mapper. Given a column u within the multi-alignment we denote as $\mathcal{L}_u \subseteq \mathcal{L}$ the subset of read occurrences that overlap that column. Furthermore, $\mathcal{L}_{u,v} = \mathcal{L}_u \cap \mathcal{L}_v$ contains all read occurrences that span both position u and position v. Inversely, $\mathcal{C}_{\mathcal{L}'}$ is used to denote the columns within the multi-alignment that are covered by *all* read occurrences $\ell \in \mathcal{L}'$.

2.2. Overview

The work flow for processing a given set of read occurrences $\mathcal{L}$ is divided into multiple stages. At first, the data is *pre-processed*, i.e. the multiple sequence alignment is optimized to avoid classification errors due to false separating columns induced by a suboptimal multi-alignment. If possible, the optimized multi-alignment is then *filtered*, i.e. areas that are not expected to contain ambiguous reads are excluded using the criteria discussed in Section 1.4.2, which reduces the number of alignment columns that have to be inspected. The remaining columns are then examined in order to identify those that fulfill the criteria for true separating columns.

In a next step, multiple such columns covering a common set of read occurrences are used to create *separation sites*. The read occurrences within such a site are then *locally classified* based on the information contained in the identified columns as associated by the separation site. Depending on the desired result of the separation these local classification results are used in different ways. One possibility is the creation of an improved multi-alignment by removing falsely mapped read occurrences based on their computed partition. If, however, the sequence of the origin of the reads is of interest, the results of the local classifications are combined if possible. This process has two purposes: firstly, the refinement of local partitions based on the information gained through other local partitions; secondly, the association of classes across multiple such local partitions to form larger classes that span wider areas of the multi-alignment. Each of the steps as well as the algorithms used in each stage are discussed in the following sections.

2.3. Pre-processing the data

Micro heterogeneity based read separation relies on the good quality of the multi-alignment produced by the read mapper. An erroneous multi-alignment can induce alignment columns that are not distinguishable from true separating alignment columns, as illustrated in Figure 2.1. Thus, prior to any analysis, the alignment should be optimized using the ReAligner [21] algorithm to reduce false over-classification originating from such suboptimal multiple alignments.

2.4. Filtering the search space

Evaluating the information content of alignment columns regarding the separation of ambiguous reads can be time consuming. Therefore it is desirable to find a search space covering all read occurrences of interest that is smaller than the entire read layout. As described in Section 1.4.2, depending on the input data there are two features within a read layout that can be used as an indicator for read ambiguity. The most obvious indicators are multiread occurrences, i.e. locations that are covered by reads that map against at least one other location. However, as already pointed out, in some scenarios we do not observe multireads, even though the data contains ambiguous reads. Another indicator that might be available in that case, is unexpectedly high coverage. Finally, for mixed sample separation or quantitative methods not even that

```
GCTAACGATCACCGCCTGT
GCTCACGATCACCGC-TGT
GCTCACGATCACCGC-TGT
GCTCACGATCACCG-CTGT
GCTCACGATCACCG-CTGT
GCTCACGATCACCGC-TGT
GCTCACGATCACCGC-TGT
GCTAACGATCACCG-CTGT
GCTAACGATCACCGC-TGT
GCTAACGATCACCG-CTGT
GCTAACGATCACCGC-TGT
GCTAACGATCACCGC-TGT
GCTAACGATCACCGC-TGT
```

Figure 2.1.: A multi-alignment with contradicting separating columns (deviations from the columns consensus are highlighted). The first separating column is induced by read ambiguity, whereas the other two separating columns are induced by different alignments of the same deletion across both variants.

indicator is available, hence no filtering is possible and every location within the layout has to be verified.

Let a *target location* be a tuple $t = (t_s, t_e)$, where t_s and t_e represent the start and end positions of a region within the read layout that must be examined for separating columns. A filter returns a set

$$\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$$

of such locations.

2.4.1. Filtering by multiread occurrences

Let $\mathcal{L}_i^m \subseteq \mathcal{L}$ be a maximally large set of overlapping multiread occurrences and $\mathcal{L}^m = \{\mathcal{L}_1^m, \ldots, \mathcal{L}_n^m\}$ the set of all such subsets contained in $\mathcal{L}$. Then the set of target locations is simply built by all tuples containing the start- and end-position of each such set:

$$\mathcal{T} = \left\{ t_i = \left(\operatorname*{arg\,min}_{l \in \mathcal{L}_i^m} l^s, \operatorname*{arg\,max}_{l \in \mathcal{L}_i^m} l^e \right), i \in \{1, \ldots, n\} \right\}. \tag{2.1}$$

Which read occurrences are multiread occurrences can be determined in $O(|\mathcal{L}|)$ time. After initially sorting the read occurrences in $O(|\mathcal{L}| \cdot \log |\mathcal{L}|)$ time the continuous groups of multireads can be determined in $O(|\mathcal{L}|)$ time, hence the overall time complexity of computing $\mathcal{T}$ is in $O(|\mathcal{L}| \cdot \log |\mathcal{L}|)$.

2.4.2. Filtering by unexpectedly high coverage

This method was suggested by *Kececioglu and Ju* and is part of their assembly correction work flow [18]. Under a simplified model one can express the probability of observing a particular depth, i.e. the number of reads that cover a given position in the read layout, using the cumulative binomial distribution function:

$$B(p,k,n) = P(K \geq k) = \sum_{k \leq i \leq n} \binom{n}{i} p^i \, (1-p)^{n-i} \, . \tag{2.2}$$

Under the assumption that all reads have the same length l and are uniformly distributed among the reference sequence, there are $g - l + 1$ possible positions for a read within the layout, where g denotes the length of the reference sequence. Thus the probability for a read to cover a given position is $\frac{l}{g-l+1}$. A particular depth within the multi-alignment can be considered unexpectedly high if

$$B\left(\frac{l}{g-l+1}, d, |\mathcal{L}|\right) \leq \chi,$$

where χ is a user-defined threshold. Using this condition, the depth threshold for columns considered overrepresented can be initially pre-computed:

$$d^* := \min_{1 \leq d \leq n} \left\{ d : B\left(\frac{l}{g-l+1}, d, |\mathcal{L}|\right) \leq \chi \right\}.$$

All read occurrences that overlap with a layout region showing a coverage higher than d^* are considered to potentially suffer from read ambiguity. Let $\mathcal{L}_i^c \subseteq \mathcal{L}$ be a maximally large set of overlapping read occurrences with each read occurrence overlapping at least one column with coverage higher than d^* and $\mathcal{L}^c = \{\mathcal{L}_1^c, \ldots, \mathcal{L}_n^c\}$ the set of all such subsets contained in $\mathcal{L}$. The target set $\mathcal{T}$ can then be constructed by building targets out of the outer limits of each subset, as previously show in Equation 2.1.

The values of the cumulative binomial distribution function can be computed in $O(\sqrt{\max\{m, n-m\}}) = O(\sqrt{n})$ time using the incomplete beta function [22] rather than evaluating the function shown in (2.2). Thus, d^* can be found in $O(d^* \cdot \sqrt{|\mathcal{L}|})$ time. After initially sorting the read occurrences by their start position in $O(|\mathcal{L}| \cdot \log |\mathcal{L}|)$ time the set of target occurrences can then be built in $O(|\mathcal{L}| + g)$ time.

2.5. Detection of separation sites

In the next stage, separating columns within the locations suggested by the filter are identified. Multiple such separating columns covering a common set of read occurrences are then grouped for their classification. This happens for two reasons: for one thing, using multiple separating columns to separate the same set of reads adds redundancy and therefore robustness against sequencing errors occurring within the separating columns; secondly, a single separating column is expected to distinguish reads between only two classes. For a group of similar regions of common origin it is unlikely to deviate from each other at the very same positions, as mutations are rare and thus not expected to occur multiple times at the same position. Therefore, each separating column is expected to be dominated by two base types, the column consensus and

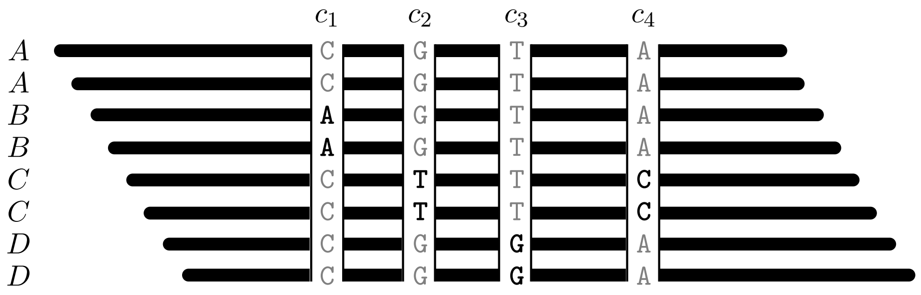


Figure 2.2.: A fraction of a read layout containing six reads belonging to four classes A, B, C and D, as denoted on the left hand side of each read. Four separating columns, c_1, c_2, c_3 and c_4 have been detected, containing deviating bases in class B (in c_1), C (in c_2 and c_4) and D (in c_3).

one deviating base type, as illustrated in Figure 2.2. A set of reads originating from k different regions can only be separated into k classes if at least $k-1$ separating columns are present, each induced by a distinct original region. Columns that separate the same classes (such as c_2 and c_4 in Figure 2.2) do, however, add the redundancy necessary to correctly classify reads that suffer from sequencing errors within separating columns. A set of separating columns $\mathcal{C}^s = \{c_1, \dots, c_n\}$ to be used for the separation of a set of read occurrences $\mathcal{L}^s = \{l_1, \dots, l_m\}$ shall be denoted as a *separation site* $s = (\mathcal{L}^s, \mathcal{C}^s)$. Due to the fact that the actual separation algorithm (see Section 2.6) is based on the Hamming distance of the strings created by concatenating the alignment characters in the columns $\mathcal{C}^s$ for each read alignment involved, all read occurrences must span all columns involved, i.e. $\mathcal{L}^s = \mathcal{L}_{c_1^s} \cap \mathcal{L}_{c^s_{|\mathcal{C}^s|}}$.

Two algorithms to differentiate between true separating columns and those containing only random correlating errors have been evaluated: the one originally used in the assembly optimization pipeline proposed by *Kececioglu and Ju* [18] which will be referred to as the *minimum support* approach, and another one developed by *Tammi et al.* [19] which will be referred to as the *pair correlation* approach. Both methods will be discussed in the following two sections. While the minimum support approach already includes the construction of separation sites as described above, a procedure to construct such sites based on separating columns detected by the pair correlation approach was added. To relax the requirements on the number of columns contained in a separation site, in Section 2.8 a method is proposed to inter-connect classification results from different neighboring separation sites and refine the partition of reads if necessary, i.e. if each of the separation sites individually did not contain the columns necessary to distinguish all present origins.

2.5.1. The minimum support approach

The minimum support method processes a multi-alignment as follows: at first, a given set of overlapping read occurrences is used to define a window within the read layout. That window undergoes a feasibility test which, if passed, states that candidate columns for separation found

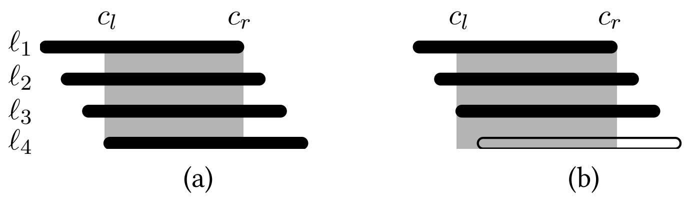


Figure 2.3.: (a) The window being searched for separating columns for a given set of read occurrences is limited to the columns that all read occurrences span. (b) The window is expanded by removing one of the read occurrences currently bounding the window.

within the window using certain criteria are statistically likely to contain true information for separation. If such columns can be extracted from a window that passed the feasibility test, a separation site is built from these columns. Each step of the procedure will be described in detail in this section.

Separation windows

Separation windows are locations within the read layout that span a certain range of alignment columns (window width) and include the reads that entirely span that range (window depth). Thus, for a given set of read occurrences $\mathcal{L}' \subseteq \mathcal{L}$ the corresponding separation window spans the columns $\mathcal{C}_{\mathcal{L}'}$ and therefore has a width of $|\mathcal{C}_{\mathcal{L}'}|$ and a depth of $|\mathcal{L}'|$. An example is illustrated in Figure 2.3(a).

Separating columns

Furthermore, criteria for *separating columns* need to be defined. A *candidate* for a separating column within a given window is a column c within that window in which a character occurs in at least two rows, but not more than $\lfloor d/2 \rfloor$ times, where d is the depth of the window. The column c is *supported* by another column d if both c and d fulfill the candidate criterion for the same pair of rows. If column c is supported by at least $t-1$ other columns, it qualifies as a separating column. The required *minimum support* t is window dependent and computed during the window feasibility test.

Window feasibility

As already pointed out in the introduction, micro heterogeneity based separation relies on the differentiation of two different kinds of deviations in read sequences: *sequencing errors* and *distinguishing base sites*, i.e. deviations that occur due to the different origins of the reads. A set of columns $c_1, \ldots, c_t$ that support each other and therefore fulfill the criteria for separating columns is considered *false positive* if none of the $2t$ errors observed originate from distinguishing base sites.

The window feasibility test consists of two stages. At first, the minimum support parameter t_{min} is chosen depending on the depth d and the width w, such that the probability of a false positive event is sufficiently low. Afterwards, it is checked whether the probability of a true positive event is sufficiently high for a window of the given dimensions and a set of t_{min} separating columns. If that is the case, the window is considered feasible.

Instead of an exact probability for the false positive and true positive events, *Kececioglu et al.* provide an upper bound for the first and a lower bound for the latter, as they expect the exact probabilities to be rather complex to determine as well as hard to compute. The upper bound for a false positive event is given as

$$f(t,w,d) := \binom{d}{2}\left(1 - \frac{1}{\binom{w}{k}}\sum_{0\leq i<t}\binom{k}{i}\binom{w-k}{k-i}\right)\cdot B\left(\frac{1}{4}\left(1-\frac{\delta}{2}\right)^{c}, t, k\right), \tag{2.3}$$

where ϵ is the sequencing error rate, δ is the rate at which distinguishing base sites occur between similar reads, $k = \lceil \epsilon w \rceil$ is the expected number of sequencing errors in a row within the window and B is the cumulative binomial distribution function as shown in equation 2.2. Furthermore, c is the estimated number of classes to be detected in the window. Since this feasibility test was originally used in sequence assembly, the authors used $c := \max\{d/r, 1\}$ as an estimate. As pointed out in Section 1.4.2, depending on the read mapping application and the input data, different coverages relative to the number of read classes are not necessarily observed. In that case, this method can only be applied with a user-supplied number of classes and if the reads of all classes are distributed equally across the reference.

Based on Equation 2.3 the minimum support for a given window shall be denoted as

$$t_{\text{min}} := \min_{t\geq 1}\{t : f(t,w,d) \leq \sigma\}, \tag{2.4}$$

where σ is a user-defined threshold. With the minimum support that guarantees the probability for a false positive event to be sufficiently low, it is now checked if the probability for a true positive is sufficiently high. Again, not an exact probability, but in this case an upper bound is provided. It is stated that the probability that a true positive separating column with support t for a given class occurs in a window with depth d and width w is lower bounded by

$$g(t,w,d) := \left(B\left(\frac{\delta}{2}, t, w\right) - B\left(\frac{\delta}{2}, \left\lfloor\frac{w}{2}\right\rfloor + 1, w\right)\right)\cdot\left(\frac{\binom{w-t}{k}}{\binom{w}{k}}, 2, min\{r,d\}\right). \tag{2.5}$$

As mentioned before, separating columns are not expected to be shared across different pairs of classes. Thus, we consider a window feasible, if

$$B\big(g\left(t_{\text{min}}, w, d\right), \lceil(1-\kappa)c\rceil, c\big) \geq 1 - \omega,$$

where κ is a copy-confidence threshold and ω is a window-confidence threshold, both specified by the user.

As mentioned in Section 2.4.2, $B(p,k,n)$ can be evaluated in $O(\sqrt{n})$ time. Furthermore, $\binom{n}{m}$ can be evaluated within machine precision in $O(1)$ time [22], thus given $f(t-1,w,d)$,

$f(t, w, d)$ can be computed in $O(\sqrt{k})$ time. Therefore, the worst-case runtime to find t^* is in $O(t^*\sqrt{k}) = O(k^{3/2}) = O(\lceil \epsilon w \rceil^{3/2})$.

Applying the same numerical techniques for the runtime analysis of $g(t, w, d)$ reveals a worst-case time in $O(\sqrt{q} + \sqrt{d}) = O(\max\{w, d\}^{1/2})$ for its evaluation. Thus, the overall runtime to check a windows feasibility is in

$$O(\lceil \epsilon w \rceil^{3/2} + \max\{w, d\}^{1.2}).$$

Enumeration of windows

Now, having a procedure for evaluating whether a window is feasible or not, the question is how to choose the windows to undergo that feasibility test. Given a set of overlapping read occurrences $\mathcal{L}'$, it is of course desirable to take as many of those occurrences into consideration as possible when performing the classification, i.e. choose a window of high depth. Since the window is built based on the common alignment columns that all read occurrences entirely span, the window with the greatest depth is also the most narrow window, i.e. the window with the smallest width. That window might not be suitable, as it might not contain sufficiently many columns to pass the feasibility test. Thus, reads are successively removed from $\mathcal{L}'$ and the feasibility test is applied on the resulting new windows.

Let $\mathcal{L}'_k \subseteq \mathcal{L}'$ with $|\mathcal{L}'_k| = |\mathcal{L}'| - k$, i.e. a set of read occurrences after removing k occurrences from $\mathcal{L}'$. For $k = 0, 1, \dots$ the subsets of $\mathcal{L}'$ are built and the corresponding windows are tested for feasibility. For each k there are $\binom{|\mathcal{L}'|}{k}$ many possible subsets of $\mathcal{L}'$. However, not each of them needs to tested for feasibility. The removal can be limited to those reads that currently bound the window, as illustrated in Figure 2.3, where the removal of ℓ_4 leads to a wider window, the removal of ℓ_2 or ℓ_3 however does not. Let $\mathcal{L}'_{k,i}$ be the set of read occurrences after removing the $k - i$ occurrences with the greatest left-ends as well as the i occurrences with the smallest right-ends from the remaining occurrences.

Lemma 1. *The reduced occurrence sets $\mathcal{L}'_{k,0}, \dots, \mathcal{L}'_{k,k}$ cover all k-reductions of $\mathcal{L}'$ that produce a window wider than the window corresponding to $\mathcal{L}'$.* □

Lemma 2. *If the set with the widest corresponding window out of $\mathcal{L}'_{k,0}, \dots, \mathcal{L}'_{k,k}$ fails the feasibility test, no feasible window can be built from a k-reduced set of $\mathcal{L}'$.* □

Due to Lemmas 1 and 2 only one window needs to be tested for feasibility for each k. The overall procedure used to reduce a given set of read occurrences until they form a feasible window is illustrated in Algorithm 1. Note that when increasing k, the corresponding subsets $\mathcal{L}'_{k,0}, \dots, \mathcal{L}'_{k,k}$ can be extended to $\mathcal{L}'_{k+1,0}, \dots, \mathcal{L}'_{k+1,k+1}$ in $O(k)$ time. Thus, the runtime of the algorithm is in $O(k'^2)$ for the stepwise reduction, where k' is the first k that reveals a feasible window. Including the runtime required for performing the feasibility test this leads to a runtime of $O(k^2 + k\sqrt{\max\{d, w\}})$ for finding a feasible window given a set of read occurrences.

Algorithm 1 Reduction of a set of overlapping reads until the remaining reads form a feasible separation window

```
Input: $\mathcal{L}' =$ subset of overlapping read occurrences
Output: TRUE if the occurrences were successfully reduced, FALSE otherwise
  for $k = 0 \dots |\mathcal{L}'| - 2$ do
      if $k = 0$ then
          $\mathcal{W} \leftarrow \{\mathcal{L}'\}$
      else
          extend $\mathcal{W} = \{\mathcal{L}'_{k-1,0}, \dots, \mathcal{L}'_{k-1,k-1}\}$ to $\{\mathcal{L}'_{k,0}, \dots, \mathcal{L}'_{k,k}\}$
      $\mathcal{L}_{\max} \leftarrow \arg\max_{\mathcal{L} \in \mathcal{W}} \texttt{width}(\mathcal{L})$
      $w \leftarrow \texttt{buildWindow}(\mathcal{L}_{\max})$
      if $\texttt{isFeasible}(w)$ then
          $\mathcal{L}' \leftarrow \mathcal{L}_{\max}$
          return TRUE
  return FALSE
```

Building separation sites

The algorithm for finding a feasible window not only reveals a set of layout columns and a set of read occurrences, but also defines the minimum support $t_{\min}$ making that window feasible, as returned by Equation 2.4. It does however not guarantee that separating columns with that support exist within the window. Such columns can however easily be detected in two iterations over all columns in the window: In the first stage, a two-dimensional table T is filled, whereas $T(i, j)$ is increased whenever a column contains the same character in rows i and j and that character occurs at most $\lfloor d/2 \rfloor$ times in that column. In the second iteration again for all columns all pairs of rows are inspected. If rows i and j agree on the same character, it occurs at most $\lfloor d/2 \rfloor$ times in that column and $T(i, j) \geq t_{\min}$, the column is reported as a separating column. That way, given a separation window, the separating columns within that window can be determined in $O(wd^2)$ time, where w and d are the width and depth of the window.

If the algorithm does not reveal any separating columns, the window is discarded. Otherwise a separation site $s = (\mathcal{C}, \mathcal{L}')$ is reported, where $\mathcal{C}$ is the set of detected columns and $\mathcal{L}'$ is the set of read occurrences that correspond to the window.

Processing a target location

As the original algorithm was integrated into the read assembly process, it would now proceed by performing the local classification on the produced separation site and then remove all overlaps between reads of different classes from the overlap graph. The procedure would then be started over again on a new layout computed from the improved overlap graph, until no further optimization is possible.

Since this work flow is supposed to be applied on mapped reads, another approach was chosen. Rather than processing each separation site individually, all separation sites found within

Algorithm 2 Extract all separation sites from a given target location

```
Input: The set of read alignments $\mathcal{L}$, a target location $t = (t^s, t^e)$
Output: A set of separation sites $\mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$
  $\mathcal{C} \leftarrow$ columns with at least two correlating deviations
        from the column consensus within $t$
  $\mathcal{S} \leftarrow \{\}$
  for $c \in \mathcal{C}$ do
      $\mathcal{L}' \leftarrow \mathcal{L}_c$
      if reduceUntilFeasible($\mathcal{L}'$) then
          if buildSeparationSite($s, \mathcal{L}'$) then
              if $|S| = \emptyset$ or $s_{|\mathcal{S}|} \neq s$ then
                  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
  return $\mathcal{S}$
```

the target location are processed subsequently. The local partitions of read occurrences resulting from the separation sites are then post processed and interconnected, which will be described in detail in Section 2.8.

Recall that every separating column has to contain at least two rows that share the same alignment character which must not occur more than $\lfloor d/2 \rfloor$ times in that column. For a given target location $t = (t^s, t^e)$, let $\mathcal{C}'_t \subseteq \{c : t_s \leq c < t_e\}$ be the set of alignment columns within the target location that fulfill that condition. For each column $c \in \mathcal{C}'_t$ the set of overlapping read occurrences $\mathcal{L}_c$ is used to find a subset $\mathcal{L}'_c \subseteq \mathcal{L}_c$ of read occurrences that form a feasible window using the previously described methods. If this is possible and sufficiently many separating columns can be found, a new separation site is reported if it is different from the previously reported site. The procedure is illustrated in Algorithm 2.

This method produces a set of distinct separation sites, where some of them might overlap with preceding ones if more valid separation sites were found in close proximity. Depending on the application the overlaps between such sites will be used to refine the partition of read occurrences and identify common origins of read occurrences that do not overlap with each other, but are connected by other overlapping read occurrences, which will be described in Section 2.8. The *minimum support* approach however has some deficiencies when used in the context of read mapping applications. Most importantly, the methods determining the significance of correlated errors require the number of classes c in a column, which is estimated using the depth at that position. As mentioned before, this is not reliable in many read mapping scenarios, particularly not if a quantitative analysis is performed. Furthermore, for the post processing methods proposed in Section 2.8 an approach that offers more control over the extent of overlap between separation sites is desirable. Therefore, another approach meant to substitute the *minimum support* method within the work flow will be discussed in the next section.

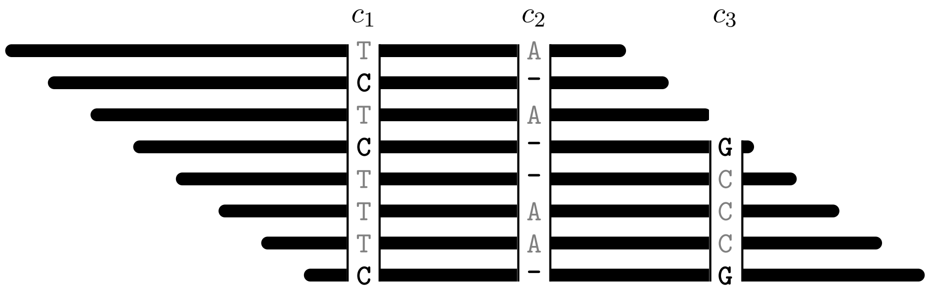


Figure 2.4.: A fragment of a read layout. Column c_1 contains $N_{c_1} = 3$ deviations from the consensus (highlighted in black), column c_2 contains $N_{c_2} = 4$ deviation from the consensus. There are $C_{c_1,c_2} = 3$ correlating deviations between the columns.

2.5.2. The pair correlation approach

The *pair correlation approach* is an alternative method to test layout columns for statistical significance regarding read separation. It does not use the intermediate model of a separation window to qualify columns for separation, but directly quantifies the probability to observe a certain kind of candidate column by chance. If it is sufficiently small, the column is considered to be reliable information for separation. There are two observations this decision is based on: the number of consensus deviations in a column and the number of such deviations coinciding between two columns, as shown in Figure 2.4.

Deviations within a column

For a given column u within the read layout, $I_{u,\ell}$ is an indicator variable denoting the event that the alignment of read occurrence ℓ deviates from the most frequent character in that column, i.e. $I_{u,\ell} = 1$ if that is the case and $I_{u,\ell} = 0$ otherwise. Then

$$N_u = \sum_{\ell \in \mathcal{L}_u} I_{u,\ell} \tag{2.6}$$

denotes the number of deviations from the consensus in column u. For a given read occurrence ℓ overlapping the layout column u, the probability that the character aligned to column u of the read corresponding to ℓ is a sequencing error[1] shall be denoted as $p_e(u, \ell)$. Since sequencing errors are considered rare events, a Poisson distribution can be assumed for the observed deviations within a column. Let n_u be the observed number of deviations in column u. The

[1]Note that the read alignment ℓ can also contain gaps, for which no sequencing error probability exists. In that case the error probability of the character left of the gap is used instead.

probability of observing n_u or more deviations can then be estimated by

$$P(N_u \geq n_u) = 1 - \sum_{i=0}^{n_u} P_{\lambda_u}(i) \tag{2.7}$$

$$\text{where } \lambda_u = \sum_{\ell \in \mathcal{L}_u} p_e(u, \ell). \tag{2.8}$$

The error probabilities p_e are provided by most sequencers in form of Phred quality scores, which were originally developed for the base calling software of the same name [23, 24].

Coinciding deviations

Let $\mathcal{L}_{u,v} = \mathcal{L}_u \cap \mathcal{L}_v$ be the set of read alignments that span both column u and column v. Furthermore,

$$C_{u,v} = \sum_{\ell \in \mathcal{L}_{u,v}} I_{u,i} \cdot I_{v,i} \tag{2.9}$$

denotes the number of coinciding deviations between the two columns. As for the number of deviations in a column, the distribution of the number of coincidences is of interest to be able to evaluate how likely they occur by chance rather than provide grounds for classification. Under the assumption that the sequencing error probability is the same at every position in the entire read layout, the coincidences are hypergeometrically distributed with parameters k, n'_u and n'_v, where $k = |\mathcal{L}_{u,v}|$ is the number of read alignments that overlap both columns and $n'_u = \sum_{\ell \in \mathcal{L}_{u,v}} I_{u,l}$ is the number of deviations observed in column u limited to those alignments (n'_v is defined accordingly). This yields the following probability to observe c correlations between columns u and v:

$$P(C_{u,v} = c) = \frac{\binom{n'_v}{c} \cdot \binom{k - n'_v}{n'_u - c}}{\binom{k}{n'_u}} \tag{2.10}$$

with $0 \leq c \leq n'_v$ and $0 \leq n'_u - c \leq k - n'_v$. If, however, instead of assuming a global sequencing error ϵ the individual error probability of each base is to be taken into account, the distribution becomes more complex to estimate. *Tammi et al.* came to the conclusion that the distribution can be estimated by:

$$P(C_{u,v} = c) = P_\lambda(c) \tag{2.11}$$

with

$$\lambda \approx \sum_{\ell \in \mathcal{L}_{u,v}} \left(\frac{n'_u p_e(u,\ell)}{p_e(u,\ell) + \lambda_u^\ell (1 - p_e(u,\ell)} \cdot \frac{n'_v p_e(v,\ell)}{p_e(v,\ell) + \lambda_v^\ell (1 - p_e(v,\ell)} \right)$$

where $\lambda_u^\ell = \lambda_u - p_e(u,\ell)$.

A test for separating columns

Finally, the two observations N and C and the previously described distributions are used to determine whether a given pair of columns consists of separating columns. Given the observed numbers of deviations $n'_u \geq d_{\text{min}}$ and $n'_v \geq d_{\text{min}}$ for two columns u and v, as well as the number of coinciding deviations c, the probability

$$\begin{aligned} p^{\text{tot}} &= p^{\text{col}} \cdot p^{\text{corr}} \\ \text{with} \quad p^{\text{corr}} &= P(C_{u,v} \geq c) = 1 - \sum_{i=0}^{c-1} P(C_{u,v} = i) \\ p^{\text{col}} &= P(N_u = n_u) + P(N_v = n_v) \\ &\quad - P(N_u = n_u)P(N_v = n_v) \end{aligned} \tag{2.12}$$

is computed, where p^{corr} is the probability to observe c or more deviations between those columns by chance and p^{col} is the probability to observe any of the two numbers of deviations n_u or n_v by chance. Furthermore, d_{min} the user-specified minimum number of consensus deviations required for a column to be considered. If

$$p^{\text{tot}} \leq p^{\text{tot}}_{\text{max}}$$

both columns are considered valid separating columns.

Building separation sites from target locations

We consider any column u within the alignment a valid separating column if it qualifies as such in any pair constellation with another column v within the alignment. To reduce the runtime for finding separating columns fulfilling that condition, it might be useful to limit the pairs of columns to those with a minimum overlap of read occurrences. Therefore, a user-specified minimum overlap parameter μ is introduced.

Lemma 3. *For any three positions u, v and w within the read layout, if $|\mathcal{L}_u \cap \mathcal{L}_v| < \mu$, then $|\mathcal{L}_u \cap \mathcal{L}_w| < \mu$ for all $u \leq v \leq w$.*

Lemma 3 implies a simple algorithm to find all separating columns within the target location as illustrated in Algorithm 3. The task that remains is to group the detected columns in order to form separation sites. Recall that a separation site can only be used to partition the read occurrences that span all separating columns that are part of the separation site. As discussed in the beginning of Section 2.5, the number of columns assembled in a separation site influences how many different read origins can be distinguished using that particular site. Also, a higher number of columns adds redundancy and error robustness. In Section 2.8 a method will be presented that allows the *interconnection* of different partitions of read occurrences resulting from different separation sites, if the sets of read occurrences of two sites intersect. The interconnection of class information allows the refinement of the partition in case a separation site

Algorithm 3 Find all separating columns within a target location

```
Input: Sorted list of columns with at least $d_{\text{min}}$ deviations $C^*$
Output: Set of separating columns $\mathcal{C}$
  $\mathcal{C} \leftarrow \{\}$
  for $i \in C^*$ do
      for $j \in C^*, j > i$ do
          if $|\mathcal{L}_i \cap \mathcal{L}_j| < \mu$ then
              break
          if $\mathcal{C} \cap \{i\} \neq \emptyset$ and $\mathcal{C} \cap \{j\} \neq \emptyset$ then
              continue
          if testColumns$(i, j)$ then
              $\mathcal{C} = \mathcal{C} \cup \{i, j\}$
```

did not contain the information necessary to detect all present origins and also allows larger partitions, i.e. building classes of read occurrences where not all members overlap, but where a path of overlapping read occurrences exists between all members.

Therefore, it is desirable to be able to control the read overlap between two separation sites, i.e. $|\mathcal{L}^{s_1} \cap \mathcal{L}^{s_2}|$ given two separation sites s_1 and s_2. The following method, which is also illustrated in Algorithm 4, uses two user defined parameters for this purpose, the *minimum site overlap* τ_{min} and the *minimum site depth* r_{min}. The set of detected separating columns within a target location is processed from the left to the right. If a previous separation site s^{pre} has been detected already, the last, i.e. the rightmost separating column c is searched, for which $|\mathcal{L}^{s^{\text{pre}}} \cap \mathcal{L}_c| \geq \tau_{\text{min}}$. That column and all previous columns c^{pre} that fulfill $|\mathcal{L}_{c^{\text{pre}}} \cap \mathcal{L}_c| \geq r_{\text{min}}$ are then grouped to a separation site. If two successive separating columns do not share any or less than τ_{min} reads, the previous separation site s^{pre} is *expanded*, i.e. more separation sites are formed by successively removing the leftmost separating column from $\mathcal{C}^{s^{\text{pre}}}$. After each removal, if the remaining columns cover a larger set of read occurrences, a new separation site is built. That way it is guaranteed that every read that is covered by a separating column will be part of at least one separation site.

Setting $\tau_{\text{min}} = r_{\text{min}} = \infty$ results in single column separation sites. The separation into more than two classes is then performed only during the interconnection phase and there is no redundancy available during the separation process. Choosing r_{min} smaller than the coverage at the processed locations produces separation sites with fewer reads but containing more columns. Setting τ_{min} to a value below the coverage results in fewer separation sites, still guaranteeing the coverage of all reads but not necessarily the usage of all separating columns if chosen rather small.

2.6. Local classification of read occurrences

Recall that a separation site $s = (\mathcal{L}^s, \mathcal{C}^s)$ is defined by a set of columns $\mathcal{C}^s$ within the read layout and a set of read alignments $\mathcal{L}^s$, which are supposed to be classified based on the

Algorithm 4 Enumerate pair correlation separation sites

```
Input: Ordered list of separating columns $C = \{c_1, \ldots, c_n\}$, read occurrences $\mathcal{L}$
Output: Ordered list of separation sites $S = [s_1, \ldots, s_m]$
  for $i = 1 \ldots |C|$ do
      $\mathcal{C}^{\text{new}} = \emptyset$
      if $|S| > 0$ then
          $s^{\text{pre}} = (\mathcal{L}^{s^{\text{pre}}}, C^{s^{\text{pre}}}) \leftarrow S_{|S|}$
          if $i < |C|$ and $|\mathcal{L}^{s^{\text{pre}}} \cap \mathcal{L}_{c_{i+1}}| \geq \tau_{\min}$ then
              continue
          $\mathcal{C}^{\text{new}} \leftarrow \{c_i\}$
          for $j = i - 1 \ldots 1$ do
              if $\mathcal{L}_{c_j} \subseteq \mathcal{L}_{c_i}$ or ( $\mathcal{L}_{c_i} \cap \mathcal{L}_{c_j} \geq r_{\min}$) then
                  $\mathcal{C}^{\text{new}} \leftarrow \mathcal{C}^{\text{new}} \cup \{c_j\}$
          if $\mathcal{L}_{c_1^{\text{pre}}} \cap \mathcal{L}_{c_i} < \tau_{\min}$ then
              expand($s^{\text{pre}}$)
      else
          $\mathcal{C}^{\text{new}} \leftarrow \{c_i\}$
          for $j = i + 1 \ldots |C|, \mathcal{L}_{c_j} \subseteq \mathcal{L}_{c_i}$ do
              $\mathcal{C}^{\text{new}} \leftarrow \mathcal{C}^{\text{new}} \cap \{c_j\}$
      if $\mathcal{C}^{\text{new}} \neq \emptyset$ then
          $S \leftarrow S + s^{\text{new}} = (\mathcal{L}_{c_1^{\text{new}}} \cap \mathcal{L}_{c^{\text{new}}_{|C^{\text{new}}|}}, \mathcal{C}^{\text{new}})$
```

characters aligned to the positions in $\mathcal{C}^s$. In this section the procedure developed by *Kececioglu and Ju* used in their repeat resolution framework is described. It computes a partition $\mathcal{P} = \{p_1, \ldots, p_n\}$, $\cup_{p \in \mathcal{P}}\, p = \mathcal{L}_s$ of the read occurrences contained in the separation site, where each pair of read occurrences not sharing the same assignment is considered to be of different origin. It provides a way to partition a separation sites reads into classes, assuring both the separation into multiple classes if the site contains separating columns of distinct origin as well as the utilization of the robustness coming with separating columns of the same origin (as discussed in Section 2.5).

Let $S_i = s_1 s_2 \ldots s_L$ be a string of length $L = |\mathcal{C}|$ were s_j is the alignment character of read alignment $\ell_i \in \mathcal{L}_s$ in column $c_j \in \mathcal{C}_s$. Under a parsimony assumption the aim is to partition $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{L}_s|}\}$ into k classes explaining the differences with as few sequencing errors as possible. Put differently, the objective is to find a partition into k classes $p_1, \ldots, p_k$ that minimizes

$$\sum_{i=1}^{k} \sum_{S \in p_i} D_H(S, T_i),$$

where T_i is the consensus sequence of class p_i and D_H is the hamming distance, which is used under the assumption that the multi-alignment of the fragments is accurate (see Section 2.3). Furthermore, if one considers that sequencing errors are rather rare events, it can additionally be assumed that within each class p one or more fragments are equal to the classes consensus. Thus the objective can be reformulated to minimize

$$\sum_{i=1}^{k} \min_{S^* \in p_i} \left\{ \sum_{S \in p_i} D_H(S, S^*) \right\}. \tag{2.13}$$

The problem stated above is equivalent to a known graph problem, the so called *k-star problem*, which is known to be NP-hard. Given an edge weighted graph G, a *star* is a set of edges that all touch a common vertex, the center of the star. The k-star problem is to reduce the graph to k vertex-disjoint stars, such that all vertices of the graph are covered and the total edge weight is minimized. The objective on the strings shown in (2.13) can be reformulated as solving the k-star problem for the complete Graph $G = (V, E, \omega)$ with $|\mathcal{S}|$ vertices and edge weights $\omega(v_i, v_j) = D_H(S_i, S_j)$.

To solve the k-star problem on complete graphs, the following integer linear program (ILP) is used and solved using a branch-and-bound approach. For the complete graph G with n vertices, there are $n^2 + n$ variables:

- for each vertex i there is a variable y_i
- for each ordered pair (i, j), including the case $i = j$, there is a variable x_{ij}.

Furthermore, the ILP contains the following $3n^2 + 3n + 1$ constraints:

$$\begin{aligned} x_{ij} &\geq 0 && \forall i, j \\ y_i &\geq 0 && \forall i \\ \sum_{i=1}^{n} x_{ij} &\geq 1 && \forall j \\ y_i &\geq x_{ij} && \forall i, j \\ \sum_{i=1}^{n} y_i &\leq k. \end{aligned}$$

All variables are restricted to integer values. The objective of the ILP is to minimize

$$\sum_{i \neq j} w_{ij} x_{ij},$$

where $w_{ij} = w_{ji} = \omega(v_i, v_j)$ is the weight of the edge connecting i and j. Given a solution to the above ILP, the solution for the k-star instance can be recovered by looking at the values assigned to each y_i – all vertices that are the center of a star are represented by the value '1' of the corresponding variable. Given the centers, the members of the star can be recovered by assigning all remaining vertices to the closest neighboring center.

With the above method there now exists a way to partition the read alignments of a target site into k classes based on the alignment characters in the assigned separating columns. However,

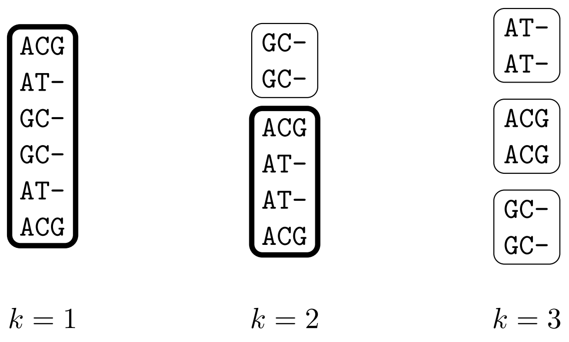


Figure 2.5.: An example for the iterative classification process for a separation site with $|\mathcal{L}| = 6$ read alignments and $|\mathcal{C}| = 3$ separating columns. The classes that still contain separating columns and thus indicate that further partitioning is required, are highlighted.

in general we have to consider the parameter k, i.e. the number of classes to be unknown. Therefore, the above procedure is repeated for increasing k starting with $k = 2$. After each iteration every class is checked for separating columns *within* the class. For that, the procedure for finding separating columns is repeated, limited to the read occurrences in each class. If no more separating columns can be found in any class, the refinement is stopped. The procedure is illustrated in Figure 2.5. Note that each column has already proven to contain statistically significant information for separation. Thus, if the pair correlation method was used, each class is simply checked for columns with at least d_{min} deviations from the consensus. If the minimum support approach is used, only the procedure to detect separating columns with a support of t_{min} is repeated.

2.7. Removing read occurrences based on consensus classification

As discussed in the introduction, given a partition of overlapping reads occurrences, depending on the application one might want to consider one class as the class of "true" read occurrences, i.e. the class containing those reads that correctly map at that location, and discard the others. For that purpose, the reference sequence is added to the separation site. Given the alignment string of the reference sequence G and a separation site $s = (\mathcal{L}_s, \mathcal{C}_s)$, s is extended to

$$s = (\mathcal{L}_s \cup \{G_{\mathcal{C}_s}\}, \mathcal{C}_s)$$

before the classification is performed, such that the reference sequence is classified together with the read alignments. As a separation site does not necessarily contain the information required to cover all different origins present in the sample (see Section 2.6), a partition of the read occurrences given after processing a separation site can only be used to decide which of them *do not* belong at that location. Which reads *do* belong at that location cannot be directly derived, since read occurrences being in the same class does not necessarily indicate a common origin.

To avoid false removals induced by single false separating columns, for each read occurrence ℓ a set of columns indicating its removal $\mathcal{R}_\ell$ is stored. Initially empty, after retrieving a partition $\mathcal{P}$ from a separation site $s = (\mathcal{L}_s, \mathcal{C}_s)$, it is updated if the read is not in the same class with the reference sequence:

$$\begin{aligned} \mathcal{R}_\ell &= \mathcal{R}_\ell \cup \mathcal{C}_s & \forall \ell \in P, \\ \text{where } P &= \bigcup_{\substack{p \in \mathcal{P} \\ G \notin p}} p. \end{aligned}$$

Finally, all read occurrences ℓ are removed from $\mathcal{L}$, for which $|\mathcal{R}_\ell| \geq u_{\min}$, where $u_{\min}$ is the user-defined minimum amount of separating columns required to remove a read occurrence.

2.8. Interconnecting classes across separation sites

As mentioned before, the classes produced from a given separation site containing error-free true separating columns guarantee that the read occurrences in different classes are of different origin, but do not guarantee that the read occurrences within the same class are of the same origin. For the original algorithm framework developed by *Kececioglu and Ju* for sequence assembly this is not necessarily problematic, as the goal is to remove false overlaps from the overlap graph produced by the assembler until such false overlaps are no longer found. Given a separation site, those edges in the overlap graph that connect reads across the classes produced by classifying the reads based on that site are considered false. If a separation site does not produce k but fewer classes, some false overlap edges remain in the overlap graph. If the information for their removal is contained in the read layout, another separation site might remove those edges at a later stage. Hence there is no need to combine the exact class information gained from multiple separation sites to a larger context. The same applies for the read mapping application, if one is only interested in the removal of false mappings, as described in Section 2.7.

If, however, a separation on a layout with “hidden” read origins as described in Section 1.4.2 is performed and the sequence of those origins is of interest, an exact and maximal classification is required as well as classes that contain more than just the number of reads that span common columns. Here a method is described to interconnect and refine multiple subsequent *local classes* produced from a single separation site to larger *global classes* that span a larger area of the read layout than a single separation site.

2.8.1. Subsequent global classification

For a given separation site $s = (\mathcal{L}_s, \mathcal{C}_s)$, the classification procedure described in Section 2.6 produces a set of disjoint classes $\mathcal{P} = \{p_1, \ldots, p_k\}$ with $\bigcup_{p \in \mathcal{P}} p = \mathcal{L}_s$. The goal is to merge

Algorithm 5 Removal of multi class assignments

```
Input: Current global partition $\mathcal{G}$, local partition $\mathcal{P}$
Output: Improved $\mathcal{G}$ containing fewer multi assignments
  for $p \in \mathcal{P}$ do
      $u \leftarrow \{\}$
      for $\ell \in p$ do
          $G_\ell \leftarrow \{g : g \in \mathcal{G}, \ell \in g\}$
          if $|G_\ell| = 1$ then
              $u \leftarrow u \cup G_\ell$
      for $\ell \in p$ do
          for $g \in G_\ell$ do
              if $g \notin u$ then
                  $g \leftarrow g \setminus \{\ell\}$
```

multiple such partitions into one global partition based on the read overlaps between the separation sites that produced the partitions. Ideally, for each connected area within the multi-alignment suffering from read ambiguity, given a set of $n = |\mathcal{S}|$ local partitions $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ a global partition $\mathcal{G} = \{g_1, g_2, \dots, g_k\}$ is recovered, where $\bigcup_{g \in \mathcal{G}} g \subseteq \mathcal{L}$ contains all ambiguous reads within that area, k is the number of different origins producing the reads that map against that range and all $\ell \in g_i$ are of common origin. The following method attempts to produce such a global partition from multiple local partitions.

The procedure assumes that the separation sites are processed as an ordered list $S = [s_1, s_2, \dots, s_n]$ in the order of the columns they contain, such that for three separation sites s_i, s_j and s_k

$$\mathcal{L}_{s_i} \cup \mathcal{L}_{s_j} = \emptyset \Rightarrow \mathcal{L}_{s_i} \cup \mathcal{L}_{s_k} = \emptyset \qquad \text{for } 0 < i < j < k \leq n. \tag{2.14}$$

Initially, there are no global classes, i.e. $\mathcal{G} = \{\}$. The local partitions resulting from each separation site are now processed sequentially. Each read that has been classified locally is assigned to one or more global classes, based on the global classes previously assigned to read occurrences sharing a local class. This is done in two stages – at first, the previous global class assignments of read occurrences are optimized. Then, in a second stage, those read occurrences without a previous global read assignment are classified globally.

The first stage starts by attempting to resolve multiple class assignments. Those occur when a previously unclassified read occurrence is locally associated with other read occurrences that have different global class assignments from previous separation sites. In that case, the unclassified read occurrences cannot be immediately assigned to a single global class. If such a read occurrence ℓ is classified again locally with other read occurrences that are consistently assigned to a global class and some of the global classes ℓ is assigned to do not occur among them, those assignments are removed as shown in Algorithm 5. Furthermore, a refinement of the global partition is required, if read occurrences that have previously been members of the same local class are now assigned to different local classes for the first time. Thus, if a global

Algorithm 6 Global partition refinement

```
Input: Current global partition $\mathcal{G}$, local partition $\mathcal{P}$
Output: Refined partition $\mathcal{G}$
  for $p \in \mathcal{P}$ do
      for $\ell \in p$ do
          $G_\ell \leftarrow \{g : g \in \mathcal{G}, \ell \in g\}$
          for $g \in G_\ell$ do
              $P_g \leftarrow P_g \cup \{p\}$
  for $g \in \mathcal{G}$ do
      if $|P_g| > 1$ then
          for $p \in P_g$ do
              $g' = \{\}$
              for $\ell \in p$ do
                  if $\ell \in g$ then
                      $g \leftarrow g \setminus \{\ell\}$
                      $g' \leftarrow g' \cup \{\ell\}$
              $\mathcal{G} = \mathcal{G} \cup \{g'\}$
```

Algorithm 7 Global class assignment

```
Input: Current global partition $\mathcal{G}$, local partition $\mathcal{P}$
Output: A global class assignment for all $\ell \in \bigcup_{p \in P} p$
  for $p \in \mathcal{P}$ do
      $G_p \leftarrow \{g : g \in \mathcal{G}, \exists \ell \in p \wedge \ell \in g\}$
      if $|G| = 0$ then
          $G_p \leftarrow G_p \cup \{g'\}$
          $\mathcal{G} \leftarrow \mathcal{G} \cup \{g'\}$
      for $\ell \in p$ do
          if $\ell \notin g \; \forall g \in \mathcal{G}$ then
              $g_p \leftarrow g_p \cup \{\ell\} \; \forall g_p \in G_p$
```

class occurs in more than one local class, those members that have been locally classified are reassigned to newly created global classes. The procedure is shown in Algorithm 6. This first stage relaxes the requirements on the separation targets – the posterior corrections of the global partition allow under-classification induced by separation sites not capable of distinguishing all classes present among the contained read occurrences (as discussed in Section 2.6) to be corrected.

Finally, in the second stage, global classes are assigned to those read occurrences that have not been classified yet at all. For that, each read occurrence ℓ with no global class assignment is added to every global class that contains read occurrences from the same local class ℓ has been assigned to, as shown in Algorithm 7.

To avoid false global classification or false global class refinement induced by single falsely

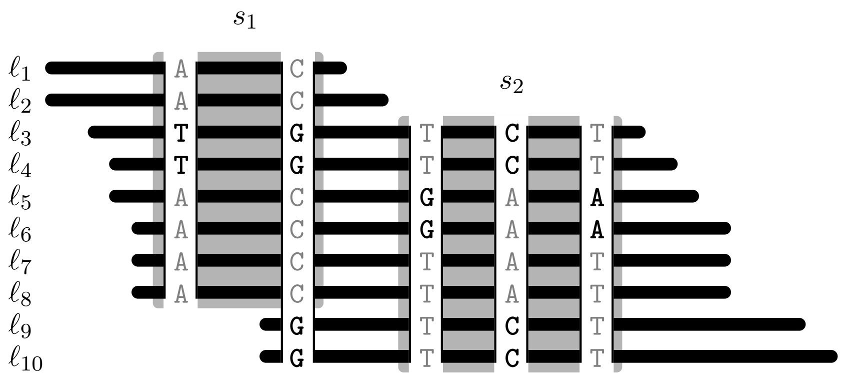


Figure 2.6.: Two separation sites, s_1 and s_2, which overlap with respect to the reads they cover. Site s_1 contains two separating columns of the same origin, thus produces only two classes. s_2 contains distinct separating columns of two different origins and thus produces three classes.

classified read occurrences, a minimum overlap parameter Δ_{min} is introduced. A global class $g \in \mathcal{G}$ is only considered in any of the previous algorithms processing a local class $p \in \mathcal{P}$ if $|\{\ell : \ell \in g \wedge \ell \in p\}| \geq \Delta_{\text{min}}$. Furthermore, as $\mathcal{G}$ contains multiple assignments in intermediate states of the algorithm, those that could not be resolved are removed after all separation sites have been processed.

Example

Consider the two separation sites shown in Figure 2.6 and assume the true classification of the read occurrences to be

$$\mathcal{G} = \{g_1 = \{\ell_1, \ell_5, \ell_6\}, g_2 = \{\ell_2, \ell_7, \ell_8\}, g_3 = \{\ell_3, \ell_4, \ell_9, \ell_{10}\}\}\,.$$

Furthermore, assume that an arbitrary enumeration procedure for separation sites has produced the subsequent separation sites s_1 and s_2 as shown in the figure. Applying the classification algorithm shown in Section 2.6 to s_1 reveals a local partition $\mathcal{P} = \{p_1 = \{\ell_1, \ell_2, \ell_5, \ell_6, \ell_7, \ell_8\}, p_2 = \{\ell_3, \ell_4\}\}$. If none of the read occurrences covered by the site has been previously classified, the following information would be the input to the procedure described above:

	ℓ_1	ℓ_2	ℓ_5	ℓ_6	ℓ_7	ℓ_8	ℓ_3	ℓ_4
global	?	?	?	?	?	?	?	?
local	1						2	

Since there are no global class assignments, the global partition is not modified in the first stage. In the second stage all read occurrences within the same local class are then assigned to a newly created global class. Now, the next separation site s_2 is processed, which reveals the local partition $\mathcal{P} = \{p_1 = \{\ell_3, \ell_4, \ell_9, \ell_{10}\}, p_2 = \{\ell_5, \ell_6\}, p_3 = \{\ell_7, \ell_8\}\}$. Thus, the input for the global classification procedure looks as follows:

	ℓ_3	ℓ_4	ℓ_9	ℓ_{10}	ℓ_5	ℓ_6	ℓ_7	ℓ_8
global	2	2	?	?	1	1	1	1
local	1				2		3	

As none of the currently observed read occurrences is assigned to more than one global class, Algorithm 5 does not perform any changes within the global partition. There are however two local classes, p_2 and p_3, containing read occurrences that are members of the same global class g_1. Thus, the global partition refinement as described in Algorithm 6 will reassign the read occurrences $\{\ell_5, \ell_6\}$ as well as $\{\ell_7, \ell_8\}$ to newly created global classes. Furthermore, read occurrences ℓ_9 and ℓ_{10} will be added to g_2 during the global class assignment, resulting in the following final association of read occurrences:

	ℓ_3	ℓ_4	ℓ_9	ℓ_{10}	ℓ_5	ℓ_6	ℓ_7	ℓ_8
global	2	2	2	2	3	3	4	4
local	1				2		3	

The final global partition produced is

$$\mathcal{G} = \{g_1 = \{\ell_1, \ell_2\}, g_2 = \{\ell_3, \ell_4, \ell_9, \ell_{10}\}, g_3 = \{\ell_5, \ell_6\}, g_4 = \{\ell_7, \ell_8\}\} .$$

Due to the insufficient information available from separation site s_1, which only allows the separation into two rather than three classes, read occurrences ℓ_1 and ℓ_2 were assigned to the same class and could not be associated with the other read occurrences of the same origin.

2.8.2. Class conflicts

To relax the requirement that separation sites can only be interconnected if processed subsequently (i.e. by processing the separation sites in the order of their columns), the previously described algorithm is extended to maintain a record of *class conflicts* $\mathcal{E} = \{\{g, g'\}, \dots\}$ containing unordered pairs of global classes that contain read occurrences that have been separated previously based on any separation site.

The knowledge about conflicting classes allows the procedure to be extended to merge global classes that occur within the same local class, i.e. a local class contains reads that are members in both global classes. Before the global class assignment (see Algorithm 7), the set of global classes G_p is checked for a subset of non-conflicting classes $G'_p \subseteq G_p$, where G'_p is created by removing all pairs of conflicting classes from G_p, as illustrated in Algorithm 8. If that procedure reveals a non-empty set, the according global classes in G_p are replaced by a new global class that contains all members of all removed classes and inherits their conflicts to other classes. Thus, when a local partition contains a class with non-conflicting previously classified reads, they are only merged when they occur in a dedicated class, i.e. the separation site contained columns corresponding to their origin.

This ability to merge parts of the global partition makes it possible to use mate pair information to produce classes that span larger areas. Now, whenever a read occurrence is assigned to

Algorithm 8 Merging global classes

```
Input: Set of global classes $G_p = \{g_1, \dots, g_n\}$, set of conflicts $\mathcal{E}$
Output: Modifies, i.e. potentially reduces $G_p$
  $G'_p \leftarrow G_p$
  for $g_i, g_j \in G_p$ do
      if $\{g_i, g_j\} \in \mathcal{E}$ then
          $G'_p \leftarrow G'_p \setminus \{g_i, g_j\}$
  if $|G'_p| > 0$ then
      $g = \bigcup_{g' \in G'_p} g'$
      for $\{g', g_x\} \in \mathcal{E}, g' \in G'_p$ do
          $\mathcal{E} \leftarrow (\mathcal{E} \setminus \{\{g', g_x\}\}) \cup \{\{g, g_x\}\}$
      $G_p \leftarrow (G_p \setminus G'_p) \cup \{g\}$
```

a global class, the corresponding mates occurrence is added to that class as well, so that they are kept synchronous. Thus, during the sequential processing of separation sites, if a global class got fragmented within the library length of two mates, the global classes might get reconnected when the occurrence of the second mate is reached.

Other than the interconnection of non-consecutive separation sites, the global class conflict information can also be used to resolve class fragmentation in the special case of separating reads of exactly two origins. In that case, every two non-conflicting classes g_i and g_j that are both in conflict with a third class g_k can be considered to be of the same origin. Depicted as a conflict graph $\mathbf{G} = (\mathcal{G}, \mathcal{E})$ containing a vertex for every global class and an edge for every class conflict, an example situation might look as shown in Figure 2.7 (up). For every global class $g \in \mathcal{G}$, from the set of global classes that are in conflict with g those pairs of classes that are in conflict with another global class in that very same set are removed and the remaining classes are merged. That way, global classes that contribute to contradicting information and would thus indicate reads from more than two distinct origins, are excluded from the merging process (in the conflict graph those classes are members of cliques of size $k > 2$). The example conflict graph after that process is shown in Figure 2.7 (down). As g_7 and g_8 are in conflict to each other, they are excluded from merging the conflicting classes of g_{10}, even though either of them could be added to $g_6 \cup g_9$ without violating any conflicts. The number of such contradicting class relations can be reduced by dropping very small classes, most importantly singleton classes before performing the previously described procedure.

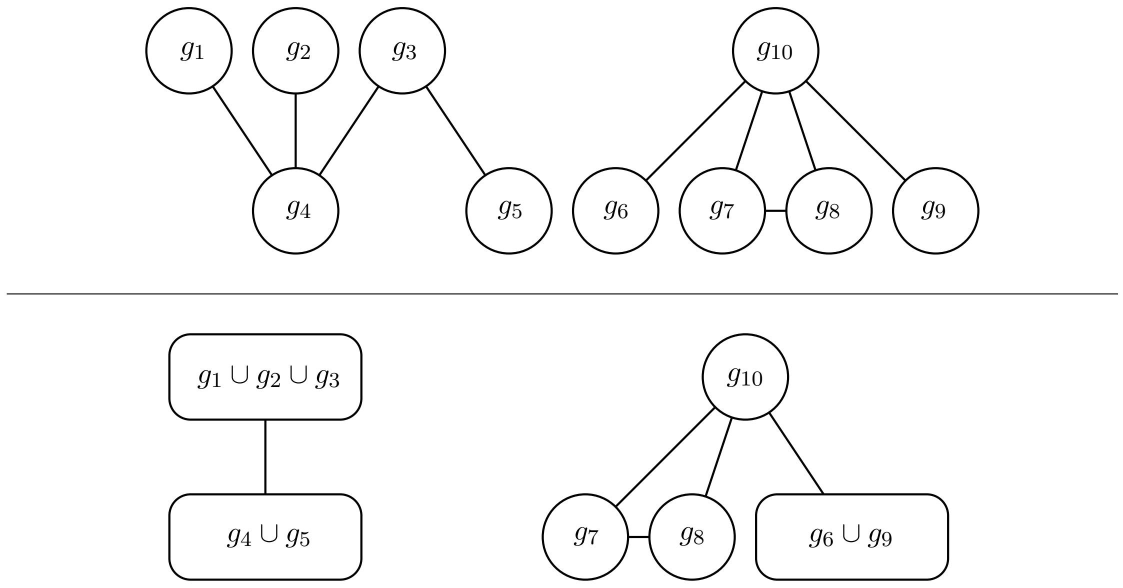


Figure 2.7.: An example conflict graph for global classes before (up) and after (down) merging classes based on common conflict partners. The clique $\{g_7, g_8, g_{10}\}$ remains unresolved, as it is contradicting, considering there are only two read origins present in the sample.

III

Evaluation

The algorithms discussed in Chapter II were implemented in C++ utilizing the SeqAn [25] library as well as the C++ bindings of the IBM ILOG CPLEX Optimizer [26]. For the evaluations, reads were simulated using Mason [27], a tool for the simulation of Illumina, 454 and Sanger sequencing processes. The read mapper RazerS [8] was then used to map the reads back against the reference sequence and to generate the multi-alignments used in the evaluations.

First of all, the sensitivity and precision of the column detection were assessed for varying sequencing errors, sequence heterogeneity rates and read lengths. Furthermore, the removal of false read occurrences was evaluated with respect to the correction of coverage profiles for quantitative methods. Those measures do not yet take into account the quality of the produced classes, as they only consider the events of a read having been classified or not (or a read having been removed or not respectively). Thus, in a next step, the homogeneity of the detected classes was quantified and compared for different numbers of classes and heterogeneity rates, extending the considerations from *whether* a read was classified or not to the *quality* of the classification.

The evaluation was then continued with a focus on the two presented application oriented post processing steps, namely the removal of falsely mapped reads (see Section 2.7) and the formation of larger, continuous classes of read occurrences (see Section 2.8).

3.1. Sensitivity of column detection

The dataset used for the sensitivity evaluation was generated based on a 500 kbp random reference sequence. Two sets of reads were generated using Mason: one set of reads $\mathcal{F}_1$ created from the entire reference sequence and a second set of reads $\mathcal{F}_2$ created only from the 250 kbp center fragment of the reference sequence, whereas heterogeneity between both read sets was induced using MASONs haplotype feature, simulating that the reads do not originate from the given sequence, but from a haplotype differing from the reference with a given haplotype SNP-rate δ. Both times MASON was configured to produce an amount of reads resulting in an average coverage of 20. The reads originating from the two sets were labeled accordingly, i.e. the reads covering only the center fragment of the reference with induced correlating errors were tagged. Based on that available information, let the *read based sensitivity* be the fraction of read occurrences originating from $\mathcal{F}_2$ that has been covered by separating columns, i.e.

$$\text{sensitivity} = \frac{|\mathcal{L}_s \cap \mathcal{L}_2|}{|\mathcal{L}_2|},$$

where $\mathcal{L}_s$ denotes the set of read occurrences that have been part of at least one separation site and $\mathcal{L}_2$ denotes the set of read occurrences originating from $\mathcal{F}_2$. The number of classes was limited to two, as only in that case the fact that a read occurrence has been classified at all (i.e. has been covered by at least one separation site) directly corresponds to the maximum class refinement (see Section 2.6). As both methods for the detection of separating columns were originally developed to be applied on reads originating from automated Sanger sequencing, the analysis was performed on reads of length 850 bp at first. Then, targeting the actual application in a read mapping context, the read length was reduced to 150 bp, which corresponds to reads produced by recent Illumina sequencing platforms. Concerning sequencing errors, MASON supports different error models for the different sequencer types. To maintain comparability, both the 850 bp and the 150 bp reads were simulated based on the *Illumina* error model, using the default settings for sequencing error probabilities $p_i = p_d = 0.001$, $p_s = 0.004$, $p_s^b = 0.003$, $p_s^e = 0.012$, where p_i denotes the probability of an insertion, p_d the probability of a deletion, p_s the mean probability of a substitution, p_s^b the probability of a substitution at the first base of the read and p_s^e the probability of a substitution at the last base respectively.

Kececioglu and Ju state that the user-specified confidence thresholds should be set to very low probabilities for their method and use $\kappa = \omega = \sigma = 0.01$ for their evaluations, thus the same values were used for this evaluation of the minimum support approach as well. For the pair correlation approach $d_{\text{min}} = 2$ and $p_{\text{max}}^{\text{tot}} = 0.001$ were used. Furthermore, $\tau_{\text{min}} = r_{\text{min}} = \infty$ was chosen to ensure that all identified separating columns are used. The read based sensitivities for different heterogeneity rates on the 850 bp reads ranging from $\delta = 0.1\%$ to $\delta = 5\%$ are shown in Figure 3.1. While the pair correlation method still detected separation sites covering almost all reads at $\delta = 0.5\%$ and still covered almost half of the necessary reads at $\delta = 0.01\%$, the sensitivity of the minimum support algorithm (which was evaluated by the authors in the context of repeat resolution with copy error rates of $\delta = 5\%$ and $\delta = 10\%$) already drastically reduced for $\delta = 2\%$. It entirely failed to detect any separating columns for $\delta = 1\%$ (which corresponds to 8.5 separating columns per read in average). Therefore, the procedure was repeated with increased confidence thresholds ($\kappa = \omega = \sigma = 0.1$), what, considering the rather high tolerance, still resulted in a rather small sensitivity for small heterogeneity rates.

The effects of different heterogeneity rates on the read based sensitivity for the 150 bp reads are shown in Figure 3.2. For this read length, only the pair correlation method revealed sepa-

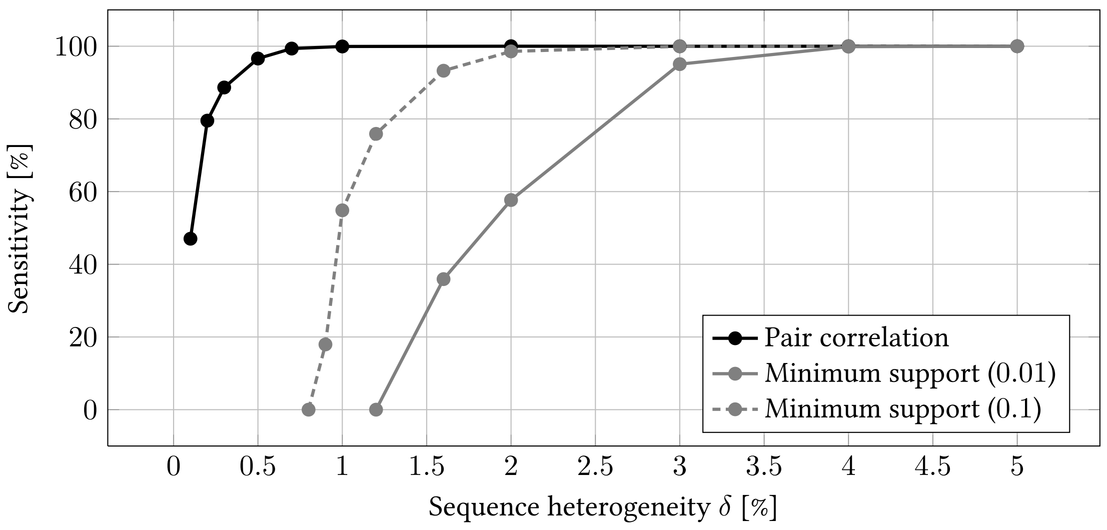


Figure 3.1.: The read-based sensitivity on the 850 bp reads for different heterogeneity rates. The pair correlation algorithm was called with $d_{\text{min}} = 2$, $p^{\text{tot}}_{\text{max}} = 0.001$ and $\tau_{\text{min}} = r_{\text{min}} = \infty$. For the minimum support method $\kappa = \omega = \sigma = 0.01$ (and $\kappa = \omega = \sigma = 0.1$ respectively) were used.

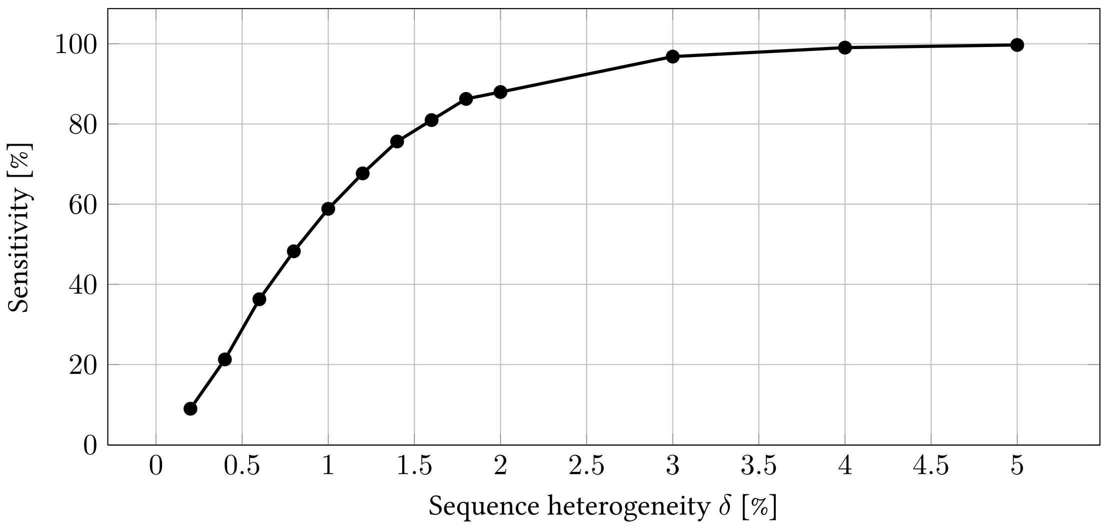


Figure 3.2.: The read-based sensitivity on the 150 bp reads for different heterogeneity rates using the pair correlation method with $d_{\text{min}} = 2$, $p^{\text{tot}}_{\text{max}} = 0.001$ and $\tau_{\text{min}} = r_{\text{min}} = \infty$.

ration sites. Used with reasonable parameters the minimum support method did not succeed to detect any separating columns. Using the pair correlation approach, with $\delta \geq 2\%$ more than 90% of the required reads were covered, whereas with $\delta \geq 0.8\%$ still more than 50% of the required reads were covered.

For all the previously described evaluation results a read based precision (as defined in the next section) $\geq 90\%$ was reached for the used parameters and simulated sequencing error rates.

As the sensitivity of the minimum support method is too low for read lengths realistic for nowadays read mapping applications and, as already discussed in Section 2.5.1, this method comes with other shortcomings making it hard to use in the context of read mapping, it will not be considered in further evaluations.

3.2. Precision of column detection

While the read based sensitivity gives an impression on what heterogeneity rate is required in order to capture sufficiently many reads under certain conditions, the focus is now shifted to the possible false positive detection of separating columns. Given the set of separation sites covering reads of different classes $\mathcal{S}_T$ and the set of separation sites covering reads of only one class $\mathcal{S}_F$, let the *read based precision* be defined as the fraction of detected columns that cover a heterogeneous set of reads:

$$\begin{aligned}\text{precision} &= \frac{|\mathcal{C}_T|}{|\mathcal{C}_T| + |\mathcal{C}_F|},\\ \text{where } \mathcal{C}_T &= \bigcup_{(\mathcal{L}^s,\mathcal{C}^s)\in\mathcal{S}_T} \mathcal{C}^s\\ \mathcal{C}_F &= \bigcup_{(\mathcal{L}^s,\mathcal{C}^s)\in\mathcal{S}_F} \mathcal{C}^s.\end{aligned}$$

Again, 150 bp reads were simulated based on a 500 kbp random reference sequence with a coverage of 20 and additional reads doubling the coverage of the 250 kbp center fragment. The heterogeneity rate between the two read classes was set to $\delta = 3\%$. This time however, the read error rate $\epsilon_{\text{tot}} = p_i + p_d + p_s$ is varied, whereas the ratio between the individual probabilities remains the same as before ($p_i = p_d = 0.25 \cdot p_s$).

The results are shown in Figure 3.3. The read based precision drops down to 87% for a sequencing error rate of $\epsilon_{\text{tot}} = 10\%$, indicating that the method is sufficiently robust to handle common sequencing error rates as they are to be expected in nowadays sequencers. The read based precision does however only take into account whether a detected column covers an area that suffers from ambiguity or not – hence the increase of false separating columns *within* these areas as well as the falsification of true separating columns is not taken into account. Therefore, in the next section the quality, i.e. the homogeneity of the detected classes will be quantified and measured for different simulation scenarios.

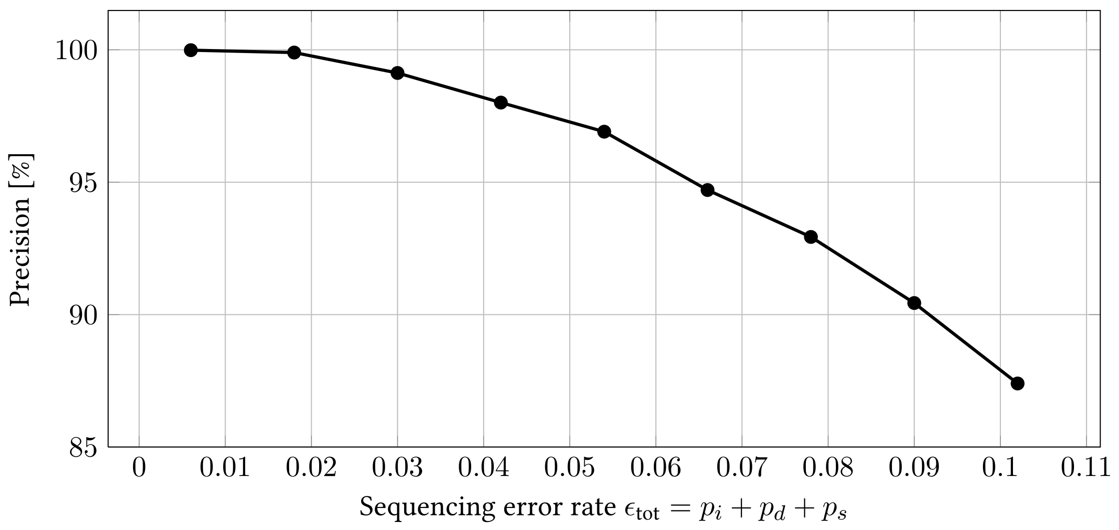


Figure 3.3.: The read-classification based precision for 150 bp reads containing two classes differing with $\delta = 3\%$ simulated with different sequencing error rates. The ratios are maintained as before, i.e. $p_i = p_d = 0.25 \cdot p_s$.

3.3. Classification quality

To evaluate the classification performance itself, given a partition of read occurrences $\mathcal{G} = \{G_1, \ldots, G_n\}$ let the *weighted mean class homogeneity* be defined as

$$h'(\mathcal{G}) = \frac{1}{\sum_{G \in \mathcal{G}} |G|} \sum_{G \in \mathcal{G}} |G| \cdot h(G),$$

where for a given class $G = \{\ell_1, \ldots, \ell_m\}$, $h(G)$ denotes the homogeneity of G, i.e. the fraction of read occurrences within G representing the largest group of read occurrences with common origin in G:

$$h(G) = \frac{\arg\max_{R \in \mathcal{R}(G)} |R|}{|G|},$$

where $\mathcal{R}(G) = \{R_1 \subseteq G, \ldots, R_n \subseteq G\}$ is a partition of G corresponding to the true origin of the read occurrences in G. To correct for the bias of perfectly homogeneous classes resulting from false separation sites covering only read occurrences of common origin, a class is only considered for this evaluation if a separation site covering reads of different origin has contributed to it, i.e. either the class itself is not 100% homogeneous or it conflicts with another class that contains read occurrences of different origin. The evaluation was performed on the same dataset as for the precision analysis, that is with a heterogeneity rate of $\delta = 0.03$ and varying sequencing error rates. The result is shown in Figure 3.4. The analysis was per-

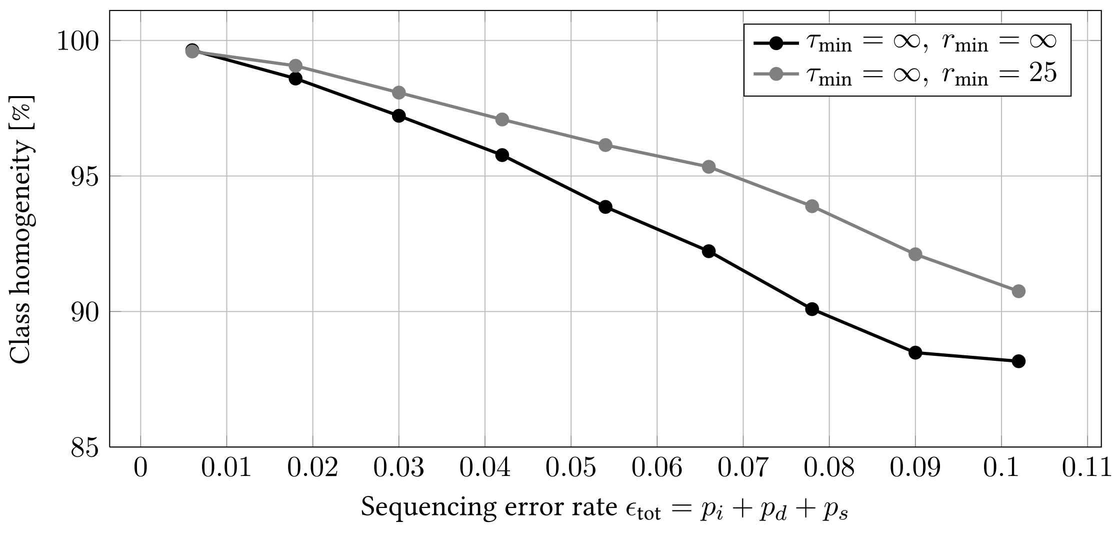


Figure 3.4.: The weighted mean class homogeneity for 150 bp reads containing two classes differing with $\delta = 3\%$ simulated with different sequencing error rates. The ratios are maintained as before ($p_i = p_d = 0.25 \cdot p_s$). One plot shows the result for single-column separation sites, i.e. no redundancy, the other one shows the result for multi-column separation sites.

formed once using $\tau_{\text{min}} = r_{\text{min}} = \infty$, i.e. with single column separation sites, and once with $\tau_{\text{min}} = \infty$, $r_{\text{min}} = 25$, producing multi-column separation sites. For the singleton sites the homogeneity drops down to 88% when the sequencing error rate reaches 10%. With multi-column separation sites an up to 3% higher homogeneity can be achieved for higher sequencing error rates.

After gaining an impression of the general impact of the sequencing error rate on the class homogeneity based on a two-class dataset, the implications of more classes being present in the sample were analysed. For that purpose the class homogeneity was observed for datasets containing reads from $2, 3, 4$ and 5 distinct origins simulated based on a 500 kbp random reference sequence. Again, each of them was simulated with an average coverage of 20. Each of the read sets was produced by setting Masons haplotype parameter to produce an error of $\delta/2$ for each origin, resulting in a pairwise heterogeneity of δ between each pair of classes. The separation method was applied using $\tau_{\text{min}} = \infty$ and $r_{\text{min}} = 30$ for all evaluations. Thus, the average number of separating columns raised with the depth and therefore with the number of read origins present in the sample. The class homogeneity analysis for the different numbers of classes was then performed varying sequence heterogeneity rates ranging from $\delta = 0.5\%$ to $\delta = 3.5\%$. The result of the evaluation is shown in Figure 3.5. Even though differences of $\delta \leq 1\%$ suffice to clearly separate two read origins when no other reads are in the sample, the class homogeneity falls down to 50% when five different read classes are present and the inter-class sequence heterogeneity is as low as $\delta = 0.5\%$. Only for heterogeneity rates $\delta \geq 1\%$

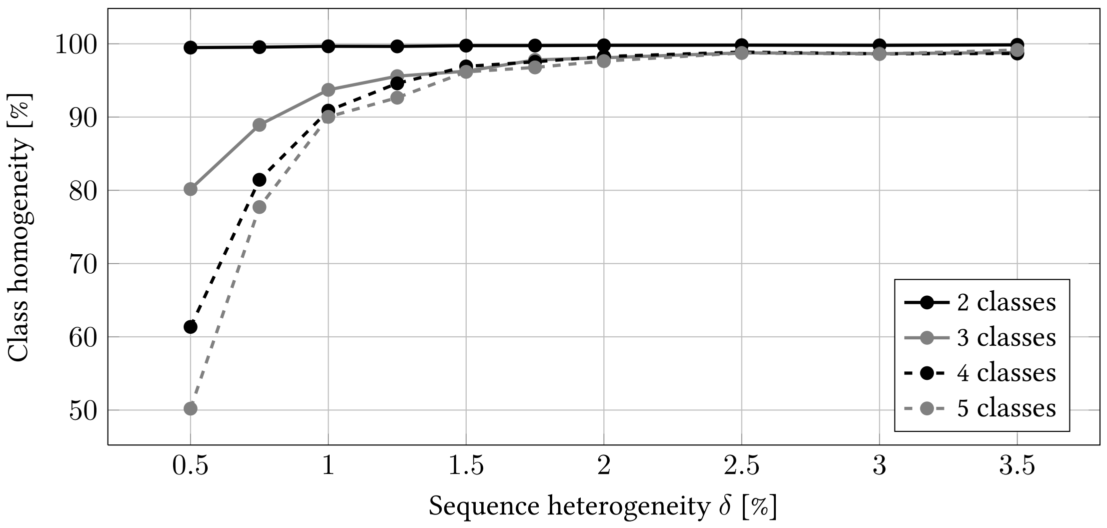


Figure 3.5.: The average class homogeneity for different sequence heterogeneity rates and different numbers of classes, i.e. different read origins.

a class homogeneity above 90% was reached in all evaluations.

3.4. Quantitative analysis

Now that an impression about the general separation performance of the previously described methods was obtained, the next analysis focuses on one of the application orientated post processing steps, namely the removal of falsely mapped reads as described in Section 2.7. For that purpose, reference sequences with five similar regions $r_1, \ldots, r_5$ of 20 kbp were simulated. Reads were then generated using Mason, producing different coverages for each distinct region, namely a coverage of 150 for region r_1, 75 for region r_2 etc., reducing the depth by 50% in each step. Finally, the simulated reads were mapped back against the reference sequence using RazerS, whereas the distance threshold for a read to be mapped was chosen such that it covers the configured heterogeneity, i.e. every read is mapped against every region. Different datasets were produced for different heterogeneity rates ranging from $\delta = 0.5\%$ to 3%.

The resulting coverage profiles are shown in Figure 3.6, where each plot illustrates the coverage profile for a different heterogeneity rate. Additionally, the original coverage profile as produced by the read mapper is shown, where every region is evenly represented. Low heterogeneity rates lead to an almost symmetric reduction of the coverage of all regions and thus do not suffice to recover how strong each region is represented within the reads relative to other regions. With an increasing heterogeneity rate the relative representations of the regions to each other become more visible, whereas regions with a high coverage are more easily recovered than regions with a low coverage. That effect is also visualized in Figure 3.7, where for each pair of regions with double (or half) coverage the pairwise coverage ratio is shown after the

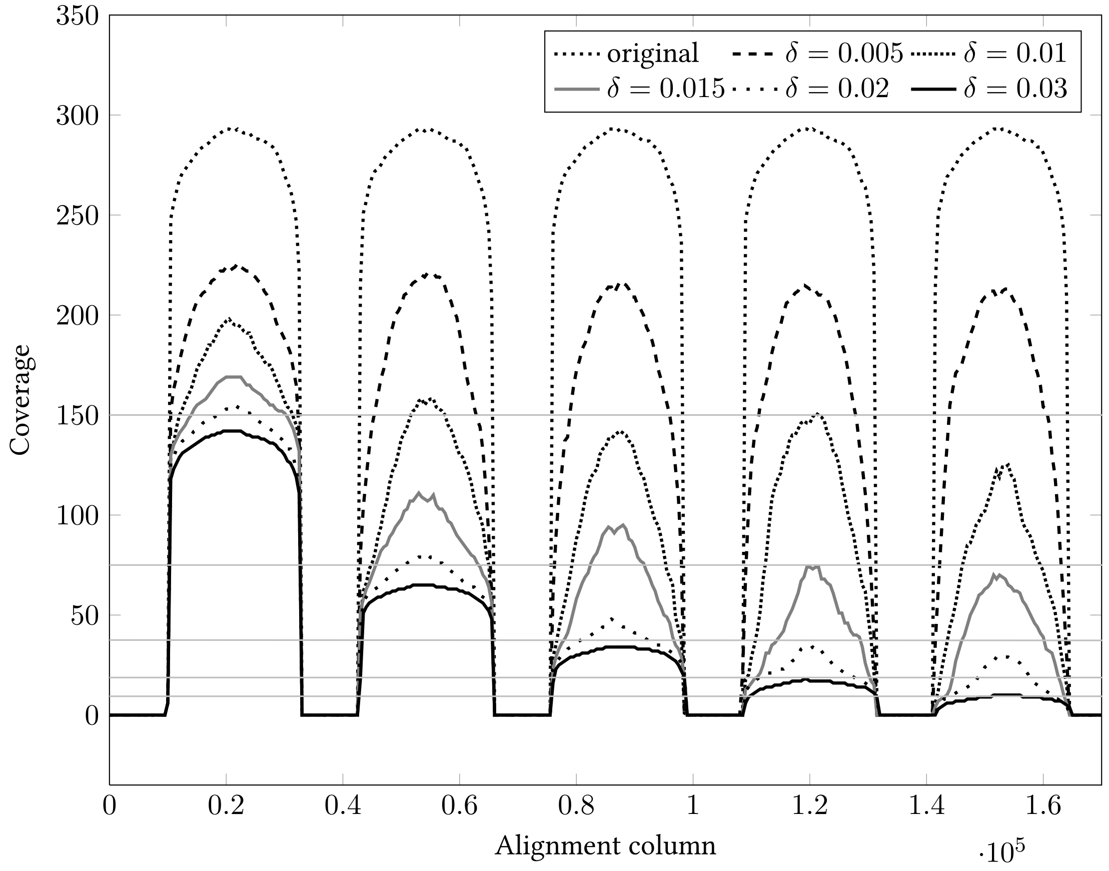


Figure 3.6.: Recovery of a coverage profile using false mapping removal. Five similar regions were generated for different heterogeneity rates. For each region reads were generated producing a different coverage for each region (150, 75, 37.5, 18.8 and 9.4, as indicated by the horizontal lines). The *original* plot shows the untreated coverage.

separation, depending on the heterogeneity rate. The regions with a higher original coverage, i.e. 150 and 75, get restored to the original pairwise ratio of 0.5 when the heterogeneity rate reaches $\delta = 3\%$, whereas the originally lowly represented regions cannot be fully recovered.

3.5. Fragmentation

Finally, also the methods for interconnecting local partitions to build larger classes of read occurrences were evaluated. For that purpose, the *class based N50* value was computed for various simulated scenarios. Let $\mathcal{P}^I = \{p_1^I, \dots, p_k^I\}$ be the partition of the input data, i.e. the true partition of the simulated reads. Furthermore, given the partition $\mathcal{G}$ computed by the separation procedure, let $\mathcal{G}_i^I \subseteq \mathcal{G}$ be the set of computed classes dominated by reads from p_i^I, i.e. $|p_i^I \cap g| \geq \frac{|g|}{2} \; \forall g \in \mathcal{G}_i^I$. Given $\mathcal{G}_i^I$, let $G_i^I = \left[g \cap p_i^I : g \in \mathcal{G}_i^I\right]$ be a sorted list of those

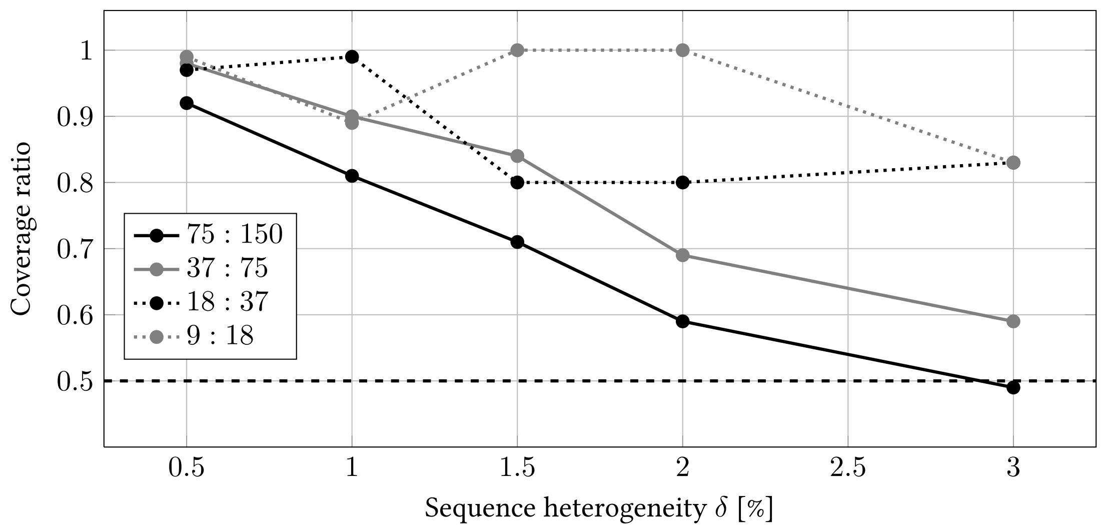


Figure 3.7.: The pairwise coverage ratios for those regions that were originally simulated with a pairwise ratio of 0.5 plotted for different heterogeneity rates after the separation.

classes limited to their dominating members, in descending order with respect to their sizes. For a given class p_i^I and $G_i^I = [g_1^i, \dots, g_m^i]$, the class based $N50$ is defined as

$$N50(p_i^I) = \frac{|g_x^i| \cdot L_f}{r_i}, \tag{3.1}$$

$$\text{with} \quad x = \min\left\{ x^* : \left| \bigcup_{j=1}^{x^*} g_j^i \right| \geq \frac{N_i^{\text{tot}}}{2} \right\}, \tag{3.2}$$

where N_i^{tot} denotes the total number of separated reads originating from p_i, L_f denotes the read length and r_i the coverage of reads originating from p_i. Given the global class g_x^i, which is the smallest global class that has to be taken into account when choosing global classes until half of all classified reads of a particular origin i are covered, the class based $N50$ value is meant to be an estimate for the number of alignment columns covered by that class within the multi-alignment. Therefore, it can be used to roughly estimate the degree of fragmentation of the class p_i^I after the separation process, as in most cases the separation process will not be able to recover an original class of reads p_i^I as a whole, but (assuming perfect class homogeneity and sensitivity) partition p_i^I into multiple classes $G_i^I = \{g_1^i, \dots, g_m^i\}$, each spanning a different range of the multi-alignment.

For the fragmentation analysis the dataset based on a 500 kbp random reference sequence containing five sets of reads with a pairwise sequence heterogeneity of δ, each producing an average coverage of 20 was used again. As the fragmentation turned out to be strongly fluctuating for small class numbers, each evaluation was performed for all possible combinations of the five simulated read classes and the average $N50$ was computed over all classes present in the sample and over all read set combinations used. For example, the evaluation for two read

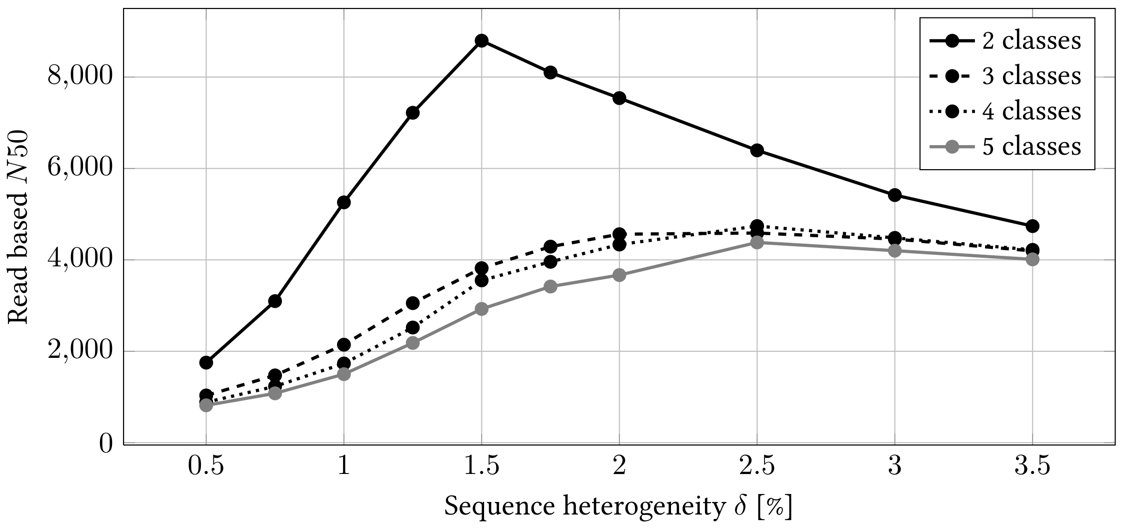


Figure 3.8.: The read based $N50$ value for different sequence heterogeneity rates and numbers of classes. The separation parameters were set to $\tau_{\min} = \infty$ and $r_{\min} = 30$ and the simulated data consisted of mate pairs with a library length of 400 bp, which were used to enhance the class sizes.

origins was performed independently for each pair of the simulated read sets and the average was built over all pairs and over both classes each time. Again, the separation method was applied with $\tau_{\min} = \infty$ and $r_{\min} = 30$. Furthermore the utilization of mate-pair information was enabled, where the simulated library length was 400 bp, thus resulting in an average gap of 100 bp between two mates. The results of the evaluation for a pairwise heterogeneity rate from $\delta = 0.5\%$ to $\delta = 3.5\%$ between the read origins are shown in Figure 3.8. For two classes, a strong increase of the $N50$ from 1700 bp at $\delta = 0.5\%$ to 8800 bp for $\delta = 1.5\%$ can be observed. The development is however reversed from that point, i.e. the $N50$ value drops down to 4700 bp when a sequence heterogeneity of $\delta = 3.5\%$ is reached. For higher class numbers that peak is not present, a value above 5000 bp is never reached. Between 3, 4 and 5 different read origins there is still a slightly higher fragmentation for higher numbers of classes, but the difference is rather marginal and decreasing with increasing δ. At $\delta = 3.5\%$ all simulations reach an $N50$ value of approximately 4500.

To get an impression on how much the mate pair based expansion of classes contributes to the N50 value and to see how much it can be improved in a two class scenario by applying the conflict based merging procedure as described in Section 2.8.2, the simulations with two classes were also performed without using the mate pair information as well as with utilizing the class conflicts to enhance the class sizes. The results are shown in Figure 3.9, which also includes the previously shown plot for two classes as a reference. Including the mate pair information to interconnect classes gives a significant increase. Without the mate pair information, the average $N50$ value reaches 1600 bp, thus remains far below the previously discussed value

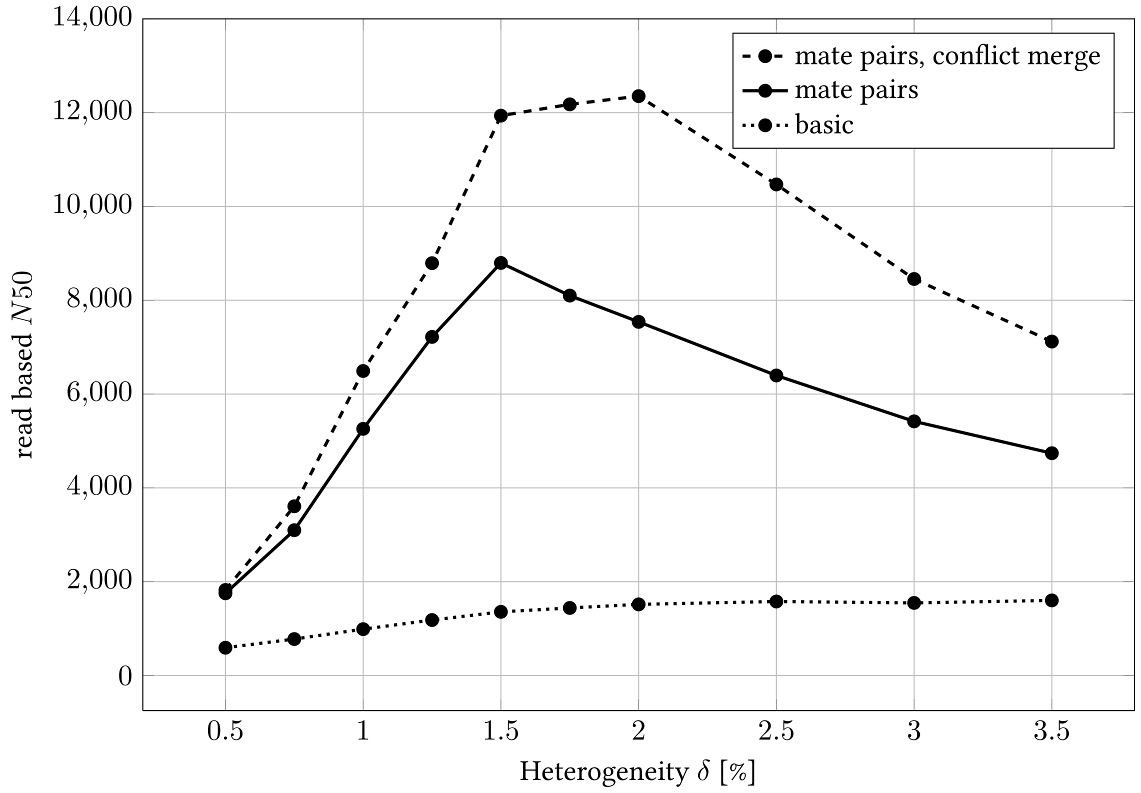


Figure 3.9.: The read based $N50$ value for different sequence heterogeneity rates when separating two classes without class merging, with mate pair based class merging and with both mate pair and class conflict based merging.

of 4500 bp, reached when classes are merged based on mate pair information. The plot also reveals that the peak behaviour of the $N50$ value for two classes seems to be induced by the method used to merge classes that are not in conflict, as it does not appear when no mate pair information is used. Using the conflicts to generate larger classes (denoted as *conflict merge* in the graph) also enhances the $N50$ value significantly. It reaches more than 12000 bp at its peak for $\delta = 2\%$ and 7100 bp for $\delta = 3.5\%$, which is almost double compared to the separation without conflict based merging. It also should be noted that no negative impact on the average class homogeneity was observed, neither when the classes were extended based on mate pair information, nor when they were extended based on class conflicts.

IV

Conclusion

In the previous chapters a work flow to separate ambiguously mapped reads in read mapping applications has been described and evaluated for different application scenarios and conditions concerning the input data. Those evaluations were performed on simulated reads of length 150 bp. One of the first conclusions to draw from these evaluations is that the minimum support approach is not suitable to separate reads of that length for any reasonable sequencing error and inter-class heterogeneity rates. This is probably due to the bounds described in Section 2.5.1, which might simply not be tight enough for reads of that length, i.e. windows of rather small width. A heterogeneity rate of 1.5% between two different read origins still produces more than two separating columns per read in average, which should allow separation to *some* degree. The minimum support method however did not reveal any of those separating columns. Both the inability to handle short reads and the requirement that the number of distinct read origins present at a certain location has to be derivable from the depth at that location make the method rather unsuitable for handling read mapping data.

When the pair correlation method is used for the detection of separating columns, $\delta = 1.5\%$ could be considered a reasonable minimum sequence heterogeneity required for any separation procedure. With that many or more correlated errors between the read origins the separation of more than 80% of the read occurrences is possible, while maintaining an average class homogeneity of more than 90% as shown in the classification quality analysis. The fact that the average class homogeneity drastically reduces when the sequence heterogeneity between the read origins drops below 1.5% for more than two classes directly corresponds to the sensitivity analysis. Since very few differences between the read origins result in few separating columns being detected, the separation sites contain only enough information to separate a subset of the present read origins. If no further refinement takes place during the interconnection phase, heterogeneous classes are the result.

Furthermore, the method has proven rather robust concerning the presence of sequencing errors, regarding both the detection of false separating columns as shown in Section 3.2 and the quality of the detected classes as shown in Section 3.3. Even with sequencing errors being present in 10% of the bases an average class homogeneity of more than 90% is maintained, which can be considered robust enough for sequencing error rates as they are to be expected in real applications.

The results of the evaluation concerning the removal of read occurrences from the multi-alignment based on local classifications produced from separation sites as shown in Section 3.4 revealed that rather many correlated errors between read classes are necessary to reconstruct different coverages as required in quantitative methods. While for two regions with a true coverage of 150 and 75 their coverage ratio can be recovered down to 0.6 with a sequence heterogeneity of $\delta = 2\%$ and to the actual ratio of 0.5 when there are $\delta = 3\%$ correlated errors between the regions, the method cannot recover almost any of the relative read quantities for regions with a simulated coverage of $9, 18$ or 37. In general, for low coverages and small differences between the read classes, methods specialized for the optimization of quantitative read mapping results or a combination of an approach based only on micro heterogeneity with any of those might be more effective.

The fragmentation analysis shown in Section 3.5 reveals, as intuitively expected, a growth in the class sizes with an increasing sequence heterogeneity between the reads of different origin. That growth however stagnates from approximately $\delta = 2\%$ on, i.e. the methods used to interconnect the results from different separation sites as presented in Section 2.8 are not able to make use of the additional separating columns induced by a higher inter-class heterogeneity rate. For two classes, after reaching a peak value rather high compared to the multi-class samples, the $N50$ value decreases rapidly for increasing heterogeneity rates. The analysis of the mate-pair and conflict-based methods for increasing the class sizes (as described in Section 2.8.2) suggests that this effect might only occur when classes are merged. When none of the methods is applied, the $N50$ value stagnates at about 1600 bp for $\delta \geq 2.5\%$ and no decrease is observed. When the mate-pair based class expansion is enabled, the previously described effect occurs, namely the $N50$ value reaches 8800 bp at $\delta = 1.5\%$ and then decreases again down to 4700 bp at $\delta = 3.5\%$. Nevertheless, it has been shown that utilizing mate-pair information can significantly decrease class-fragmentation, leading to up to five times larger $N50$ values when two classes are separated. Also the conflict-based class expansion for two classes has proven effective, increasing the $N50$ value by up to 50% compared to the values achieved when using only the mate-pair-based approach.

Possible future improvements of the presented methods therefore certainly include enhancing the inter-connection method for local partitions. More inter-class sequence heterogeneity should produce larger continuous classes of reads. For $\delta = 3\%$ there are in average 9 separating columns present within each mate-pair, a number of anchor points that should suffice to distinguish rather large continuous areas within the multi-alignment.

Bibliography

[1] M.L. Metzker. Sequencing technologies—the next generation. *Nature Reviews Genetics*, 11(1):31–46, 2009.

[2] F. Sanger, S. Nicklen, and A.R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463, 1977.

[3] M.L. Metzker. Emerging technologies in dna sequencing. *Genome research*, 15(12):1767, 2005.

[4] Wetterstrand KA. Dna sequencing costs: Data from the NHGRI large-scale genome sequencing program. `http://www.genome.gov/sequencingcosts/` [Accessed 2011-08-20].

[5] J.C. Venter, M.D. Adams, E.W. Myers, P.W. Li, R.J. Mural, G.G. Sutton, H.O. Smith, M. Yandell, C.A. Evans, R.A. Holt, et al. The sequence of the human genome. *science*, 291(5507):1304, 2001.

[6] Daniel Huson. Lecture notes 'Bioinformatics I', 2010. Available online at `http://ab.inf.uni-tuebingen.de/teaching/ws09/bioinformatics-i/11-assembly.pdf`.

[7] John Gallant, David Maier, and James Astorer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50 – 58, 1980.

[8] D. Weese, A.K. Emde, T. Rausch, A. Döring, and K. Reinert. RazerS-fast read mapping with sensitivity control. *Genome Research*, 19(9):1646, 2009.

[9] S. Burkhardt, A. Crauser, P. Ferragina, H.P. Lenhof, E. Rivals, and M. Vingron. q-gram based database searching using a suffix array (quasar). In *Proceedings of the third annual international conference on Computational molecular biology*, pages 77–83. ACM, 1999.

[10] K.R. Rasmussen, J. Stoye, and E.W. Myers. Efficient q-gram filters for finding all ε-matches over a given length. *Journal of Computational Biology*, 13(2):296–308, 2006.

[11] D.R. Bentley. Whole-genome re-sequencing. *Current opinion in genetics & development*, 16(6):545–552, 2006.

[12] S.B. Ng, E.H. Turner, P.D. Robertson, S.D. Flygare, A.W. Bigham, C. Lee, T. Shaffer, M. Wong, A. Bhattacharjee, E.E. Eichler, et al. Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, 461(7261):272–276, 2009.

[13] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.

[14] A. Barski, S. Cuddapah, K. Cui, T.Y. Roh, D.E. Schones, Z. Wang, G. Wei, I. Chepelev, and K. Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129(4):823–837, 2007.

[15] J. Handelsman. Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4):669, 2004.

[16] D.H. Huson, A.F. Auch, J. Qi, and S.C. Schuster. MEGAN analysis of metagenomic data. *Genome research*, 17(3):377, 2007.

[17] A. Mortazavi, B.A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628, 2008.

[18] John Kececioglu and Jun Ju. Separating repeats in dna sequence assembly. In *Proceedings of the fifth annual international conference on Computational biology*, RECOMB '01, pages 176–183, New York, NY, USA, 2001. ACM.

[19] M.T. Tammi, E. Arner, T. Britton, and B. Andersson. Separation of nearly identical repeats in shotgun assemblies using defined nucleotide positions, DNPs. *Bioinformatics*, 18(3):379, 2002.

[20] B. Li, V. Ruotti, R.M. Stewart, J.A. Thomson, and C.N. Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493, 2010.

[21] E.L. Anson and E.W. Myers. ReAligner: a program for refining DNA sequence multialignments. *Journal of Computational Biology*, 4(3):369–383, 1997.

[22] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C*, pages 178–180. Cambridge Univ. Press Cambridge, 1992.

[23] B. Ewing, L.D. Hillier, M.C. Wendl, and P. Green. Base-calling of automated sequencer traces using Phred. I. accuracy assessment. *Genome research*, 8(3):175, 1998.

[24] B. Ewing and P. Green. Base-calling of automated sequencer traces using Phred. II. error probabilities. *Genome research*, 8(3):186, 1998.

[25] A. Döring, D. Weese, T. Rausch, and K. Reinert. Seqan an efficient, generic C++ library for sequence analysis. *BMC bioinformatics*, 9(1):11, 2008.

[26] IBM Software. IBM ILOG CPLEX Optimizer. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/` [Accessed 2011-09-10].

[27] Manuel Holtgrewe. Mason - a read simulator for second generation sequencing data. *Fachbereich Mathematik und Informatik Tech Rep*, TR-B-10-06, 2010.

List of Figures

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit

Handling ambiguity in read mapping applications

selbstständig angefertigt habe. Alle von mir verwendeten Quellen wurden angegeben.

Berlin, den 20.10.2011 Sven-Leon Kuchenbecker