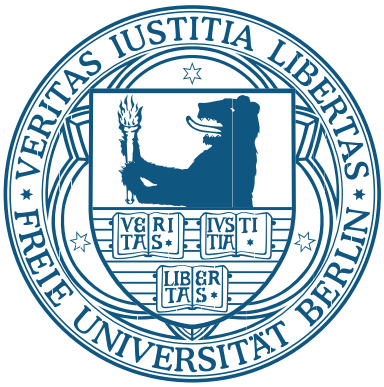
Multi-Split-Mapping of NGS reads for variant detection



Masterthesis

Master of Science Bioinformatics

Fachbereich Mathematik und Informatik
Freie Universität Berlin

Kathrin Trappe

March 6, 2012

Supervisors: Prof. Knut Reinert
Dr. Tobias Rausch
Anne-Katrin Emde

Declaration of Originality

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the work of others has been acknowledged.

Kathrin Trappe, March 6, 2014

Destiny was funny stuff, he knew. You couldn't trust it. Often you couldn't even see it. Just when you knew you had it cornered, it turned out to be something else – coincidence, maybe, or providence.

–Terry Pratchett, *Wyrd Sisters* (1989)

Contents

Abstract

Read mapping is a fundamental task in DNA sequence analysis. Current read mapping tools are very fast and precise but usually fail to map reads that cross breakpoints of structural variants (SVs), or exon-exon junctions in the case of RNA-sequencing. These breakpoints cause one or even multiple splits in the read-to-reference alignment, with parts of the read mapping to different locations on the reference sequence. Identification and classification of SVs is important to evaluate their functional impact but remains challenging. So far, there is no sophisticated SV detection method that can determine all types of SVs at single-nucleotide resolution while being independent from different platforms like Illumina or 454, or from paired-end and single-end reads.

We designed and implemented a sound generic multi-split chaining method using the C++ library *SeqAn* that uses SeqAn's exact local aligner *Stellar* to detect splits of a read. Compatible local matches of a read are then identified, and all compatibility information is stored in a *split-read graph* representation of the matches. We then use a DAG shortest path algorithm to determine the most probable chain of splits, and report the underlying breakpoints.

Our approach is more versatile compared to existing split-read methods. It allows for multiple splits at arbitrary locations in the read, and is able to detect inversions, inter- and intra-chromosomal translocations, duplications, insertions, and deletions. At the same time, it is independent of the read length. We successfully applied our method to simulated Illumina read data and also to 454 RNA-Seq data, yielding robust results that can compete with the results of the tool SVDetect and the Illumina and the 454 analysis software.

1. Introduction

Since the first sequencing of the human genome in 2001 (Venter et al. (2001), Lander et al. (2001)), the possibilities for DNA sequence analysis and with it the insight into genomics and molecular biology has increased manifold. Further advances in sequence technologies (*Next-generation sequencing* (NGS)) brought with it fast and cheap high-throughput of sequence data, making it available to a large amount of research facilities. One of the many applications in sequence analysis is the detection of structural variants (SVs) in resequencing. SVs are large genomic rearrangements, e.g. inversions, duplications, and indels, and are often involved in cancer diseases (The 1000 Genomes Project Consortium, 2010).

Defining the exact location of a SV down to single-nucleotide *breakpoint* resolution is an important requirement for classifying SVs, evaluating their impact, and reconstructing personal genome sequences (Abyzov and Gerstein, 2011). Knowledge about location and classification of SVs is a prerequisite for understanding the functional impact of these SVs in terms of diseases, and possible drug development.

However, detection and characterisation of SVs is still challenging and difficult (Alkan et al., 2011). SVs vary widely in size and complexity, and tend to reside within repetitive DNA. Until now, no human genome has been published for which the whole spectrum of SVs has been resolved. Due to their complexity, SVs are hard to detect using standard read mapping tools.

Read mapping is one of the fundamental tasks in sequence analysis, and reveals information about the origin of the sequenced reads and the sequence content (Langmead et al. (2009), Li and Durbin (2010)). In the ideal case, the whole read maps against the reference genome providing exact sequence information about the donor genome. With the advance of NGS and increasing read length, the reads become more likely to cross so called breakpoints in the case of SVs, or exon-exon junctions in the case of RNA-Seq. This prevents the mapping of the whole read (Li and Homer, 2010). Instead, parts of the read map to different locations in the reference.

As a result, a variety of computational and experimental methods for SV detection have emerged during the last years. Typically, each of them focuses on a specific type of SV. However, application of different tools to the same human DNA dataset shows only a low level of overlap, resulting in biases in global discovery (Alkan et al., 2011). This fact shows the need for a more generic SV detection approach which is able to detect a wider range of SV types, but at the same time is still able to resolve the exact breakpoint resolution. One of the most promising general approaches is *split-read* mapping since it can detect SVs at single-nucleotide resolution (Alkan et al., 2011).

The current approaches for split-read mapping usually divide the read into two or three, sometimes overlapping, parts and then use a fast read mapper like, e.g., Bowtie

(Langmead et al., 2009) or BWA (Li and Durbin, 2010) to find the split-read mappings (Au et al. (2010),Zhang and Wu (2011)). There are often additional constraints on collinearity, maximal allowed distance between matches, or the location of matches (e.g. Emde et al. (2012)). These existing approaches are rather rigid and may miss breakpoints from more complex SVs like inversions or inter-chromosomal translocations. One way to cope with this is to successively split the read in two parts of different lengths (given a minimal required length for each part) (Zhang and Wu, 2011).

Besides SplazerS (Emde et al., 2012), all existing split-read tools are restricted to either single or paired-end data. All split-read methods allowing for all SV types only support paired-end data (Xi et al., 2011). Often, they only support a certain sequencing platform such as Illumina or 454. Also, SplazerS and Pindel (Ye et al., 2009) are the only methods so far that determine the exact breakpoint location.

In our approach, we aim to overcome the problem of artificial splits by using a local match approach on the whole read instead of the global match approach of artificial splits of the read as the existing approaches do. This allows for splits at arbitrary locations within the read. In addition, we do not set constraints on the locations of the matches in the reference and thereby allow for all types of SVs including inversions, inter- and intra-chromosomal translocations, duplications, insertions, and deletions. To our knowledge, our method is the first one supporting single-end reads while being able to detect multiple SV types, but it is still extendible to support paired-end data. At the same time, our method is independent from the sequencing platform. By using a similar approach to AGE (Abyzov and Gerstein, 2011) that is implemented in SplazerS, we determine the precise breakpoint location at single-nucleotide resolution.

We use *Stellar (Swift Exact LocaL AligneR)* (Kehr et al., 2011), which is implemented in the SeqAn library (Döring et al. (2008), Gogol-Döring and Reinert (2009)), as a local aligner. Stellar is a fully sensitive, exact local pairwise aligner that guarantees to report all matches of a given minimal length and maximal error rate ϵ. Using Stellar, we allow for multiple breakpoints at variable positions both in the read and in the reference. Thereby, we are guaranteed to find all split matches that conform to a specified minimal local match length and maximal error rate criterion.

To find the most probable combination of matches for each read, all matches of one read are checked for *compatibility*. Roughly, we define compatibility as an overlap of matches either within the reference (in the case of insertions) or read (as for transpositions, duplications, deletions and inversions) sequence. All match compatibility information is stored in a *split-read graph* representation of the matches. The graph contains artificial start and end vertices, and each path from start to end represents a possible multi-split-match or *chain* of the read. The orientation of matches can be arbitrary which allows for inversions. All chains with a minimal score (or maximal edit distance) can be obtained through graph algorithms. In the end, all SVs supported by at least a specified number of reads will be reported.

1.1. Aims and Goals/Objectives

Our goal is to detect all occurring splits or breakpoints within any read, and thereby detect structural variants as named above. We designed and implemented a method using the C++ library SeqAn that (1) generates local matches of reads using Stellar, (2) chains these local matches into *multi-split-matches*, localizing optimal junction positions, and (3) outputs all multi-split-matches, i.e., chains, that obtain a specified minimal score. As an optional step (4), we call SVs (or novel RNA-Seq junctions) supported by multiple reads.

1.2. Thesis Structure

First, we explain the necessary research context including NGS, read mapping and split-read mapping in Chapter 2. Approaches and methods based on split-read mapping are reviewed in Chapter 3. We introduce our method and our test datasets including assembled contigs of simulated Illumina reads and real 454 human RNA-Seq reads in Chapter 4. Results and comparison to Illuminas Grouper pipeline and SVDetect, and the 454 analysis pipeline are presented and discussed in Chaper 5.

2. Research Context

In this chapter, we shortly review the fields of NGS, read mapping, structural variations, and the concept of split-read mapping. More details to different approaches and previous work in the field of split-read mapping is given in Chapter 3 Related Work. We start by reviewing read mapping and NGS in the following section. Next, we explain SVs and their impact in more detail in Section 2.2. Split-read mapping is explained in more detail in Section 2.3.

2.1. NGS and Read Mapping

With the discovery of the DNA double helix structure in 1953 (Watson and Crick (1953), Franklin and Gosling (1953)) came the ultimate goal of deciphering the human genome. The first major step towards this goal was made when the two competing human genome projects (Venter et al. (2001), Lander et al. (2001)) announced the first sequencing of a whole human genome in 2001. However, sequencing was expensive and laborious with a low throughput which gave the need for a better sequencing technology (Schuster, 2008).

Next-generation sequencing (NGS) allows to produce an enormous volume of data (giga base-pairs (Gbp) per machine per day) at low cost, making it also available to a greater number of research facilities (Li and Homer, 2010). NGS has therefore revolutionised DNA sequence analysis, and thereby had a huge impact on scientific approaches in medical and molecular biology research (Metzker, 2010).

NGS also brings more advantages regarding the quality of the sequence data. Sequence data are less biased (Alkan et al., 2011), and therefore have a potential for revealing the whole spectrum of genetic variation. The most important benefit is that it is possible to detect a large variety of variant classes with one sequencing experiment.

Different sequencing technologies exist including Illumina and 454, Helicos and SOLiD, and their sequence data have different properties and advantages. For example, Illumina reads have a short sequence length (35 bp to 100 bp) and a low indel error rate. Both Illumina and 454 reads have a high base quality towards the 5'-end. Helicos reads are also short and have a low substitution error rate (Li and Homer, 2010). Many sequence data analysis tools exploit certain advantages making them also specific to these technologies. The scale and resolution of these analysis tools have increased manifold. Applications include, e.g., genome-wide variation scan, analysis of transcriptomes (RNA-Seq), and the assembly of new genomes or transcriptomes (Li and Homer, 2010). With the huge amount of data of NGS also comes the need for more efficient and at the same time accurate analysis methods and tools.

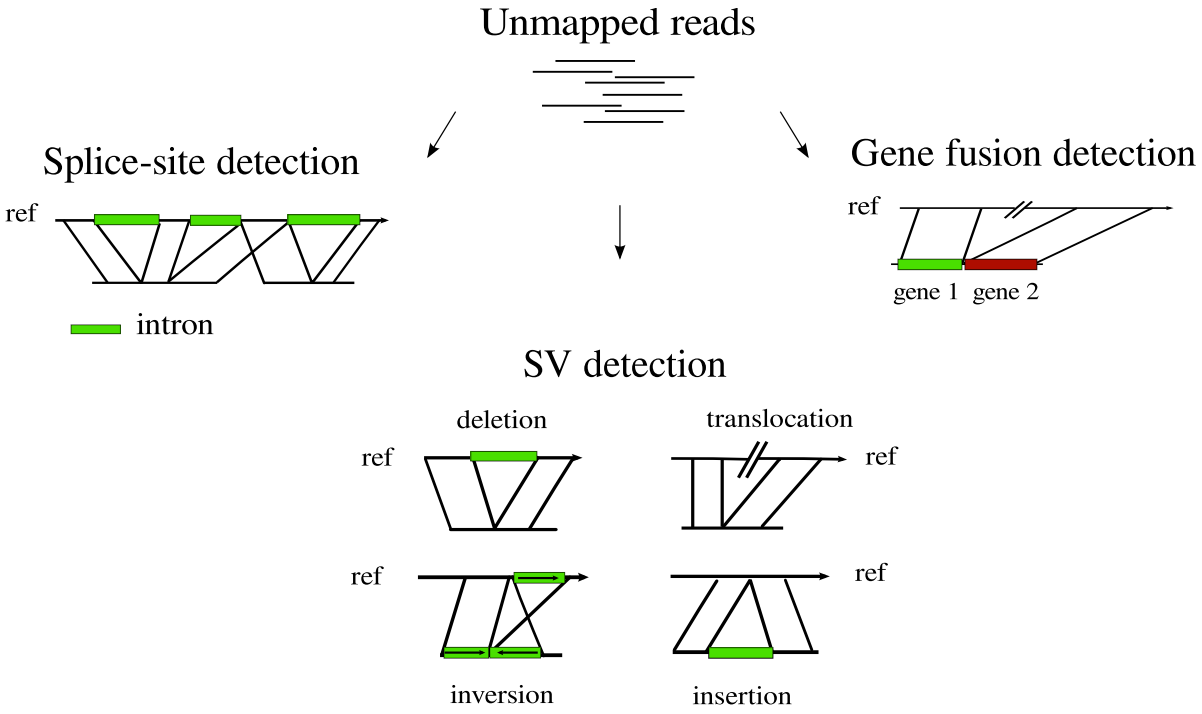


Figure 2.1.: Split-read detection possibilities of unmappable reads: Split-read approaches can be applied to splice-site detection, SV detection or gene fusion detection.

One of the fundamental problems in bioinformatic NGS data analysis is read mapping. In the standard case, for a given set of read sequences R, a reference sequence G, and a distance $k \in N$ (or error rate ϵ), the goal is to find all sub strings g of G that are within distance k (or error rate ϵ) to a read $r \in R$. Occurrences of such substrings g in G are then called matches. Common distance measures for k are edit distance, where mismatches and indels are allowed, and Hamming distance, which only allows for mismatches. Searching for matches of each $r \in R$ is computationally expensive. Therefore, most read mapping tools consist of two steps. First, candidate regions are identified using a filtration approach usually based on an index structure of either the reads or the reference sequence, or both. The candidate regions are then validated in a verification step. Enhancements in either step have produced fast, space efficient and at the same time sensitive standard read mapping tools (Li and Durbin (2010), Langmead et al. (2009), Weese et al. (2009)).

We will also concentrate on initially unmappable reads since we focus on breakpoint detection. This will reduce the amount of data we have to process. With the advance of NGS technology and increasing read length, reads are more likely to cross so called *breakpoints* arising from structural variants (SVs) (Li and Homer, 2010) or exon-exon junctions (see Figure 2.1). These *split-reads* then map partially to different locations and cannot be detected by standard read mapping. We explain SVs and their impact in more detail in the following section. Split-read mapping is introduced in Section 2.3.

2.2. Structural Variants

Single-nucleotide polymorphisms (SNPs) were long thought to be the main source of human genome variation. Already roughly 3.3 Million single-nucleotide variants were identified when the genome of Watson was compared to the reference genome (Metzker, 2010). However, more recent comparison of human genomes show that base pair alteration is more likely to occur as a consequence of SVs rather than of SNPs (Alkan et al., 2011). They also have a more significant impact on phenotypic variation than SNPs (Tuzun et al., 2005). Compared to SNPs, SVs affect up to several hundred base pairs.

Originally, SVs were defined as insertions, deletions and inversions greater than 1 kb in size (Feuk et al., 2006). Later the definition included copy number variants (CNVs) and translocations (Stankiewicz and Lupski, 2010)). After NGS technologies and widely (re)sequencing attempts of human genomes (The 1000 Genomes Project Consortium, 2010) led to an increase in discovery of smaller SV events, the definition changed to all sequence variants other than single-nucleotide variants (Metzker, 2010). We refer to this general definition throughout this thesis when using the term SV.

SVs can be classified into insertions and duplications, inversions, deletions and inter- and intra-chromosomal translocations (see Figure 2.1), and differ widely in their size. Insertions and duplications correspond to a gain of sequence content compared to the reference, deletions to a loss (marked green in SV Detection in Figure 2.1). For inversions, some part of the sequence is reverse-complemented to the reference (marked green with black direction indicator). For translocations, parts of the sequence match to different locations within the same chromosome (intra-chromosomal), or within another chromosome (inter-chromosomal). In the case of RNA-Seq, splice junctions produce a pattern similar to genomic deletions, and gene fusions a pattern corresponding to translocations. Indels of 1-10 bp are conventionally referred to as micro-SVs and over 1 kb as large ones (Zhang et al., 2011). Also, SVs tend to arise in repetitive regions, making the surrounding sequence information ambiguous.

The amount of studies and development in SV discovery is huge, meeting the fact, that SV discovery and genotyping is crucial to understand many common and severe diseases. The most extensive study on genomic variation in human genomes to date is the 1000 Genomes Project (Mills et al., 2011). However, there is still need for improvement due to the complexity of SVs (Alkan et al., 2011).

There are several sequencing-based methods for discovering SVs (Medvedev et al., 2009). Different studies focus on different applications or approaches, or include different available data. For instance, Sebat et al. (2007) is the first study to report CNVs in a common complex neuropsychiatric disease. Tuzun et al. (2005) were the first implementing a paired-end sequencing approach for SV studies. Other common approaches besides using paired-end information are read depth analysis, assembly methods and split-read methods. All four approaches have different advantages and disadvantages depending on the SV type and the SV surrounding sequence structure (Alkan et al., 2011).

Assembly methods are mainly used to reconstruct novel insertions or duplications and are computationally expensive. Read-depth approaches make use of the difference

in the number of reads mapping to a particular location (CNV) due to duplications and deletions, and can therefore detect only losses (deletions) and gains (duplications) (among others Mills et al. (2011), Tuzun et al. (2005)).

Other methods make use of paired-end information. Paired-end reads are sets of pairs of reads with a known distance between the two pair members. Normal mapping paired-end reads remain in the known distance to each other. Therefore, SV discovery methods using paired-end data make use of discordances in the read distance. However, the paired-end method will not work if one of the reads is crossing a breakpoint (see also Section 2.3). Pairs with only one read mapping to the reference can be used to at least locate possible SVs. In the case of insertions larger than the insert size of the reads, the power of these algorithms is limited (Xi et al., 2011).

Paired-end based algorithms are usually either clustering-based or distribution-based, and are therefore more suited for either large or small SV or indel detection, respectively. The tool BreakDancer (Chen et al., 2009) is a Perl/C++ package that consists of two complementary programs. BreakDancerMax predicts five types of larger structural variants: insertions, deletions, inversions, inter- and intra-chromosomal translocations using short paired-end reads that are mapped with unexpected separation distances or orientation. BreakDancerMini detects small indels. In combination, they are able to detect a wide range of SVs of different types and size.

Our method is based on a split-read approach. We explain split-read mapping in the following section.

2.3. Split-read Mapping

For split-read mapping, the original read mapping problem is altered as follows: Instead of finding matches of the whole read $r \in R$, we are looking for a split-read match $m_1 m_2 ... m_n$ of r with $m_1 \prec m_2 \prec ... \prec m_n$ according to the read position. For n split parts there are $n-1$ splits. The reference position of each m_i can be arbitrary. In other words, the alignment of r to the genome is broken in two or more split parts (Alkan et al., 2011). A continuous stretch of gaps in the alignment between m_i and m_j in the read indicates a deletion, gaps in the reference indicate an insertion. Translocations and inversions are similar to the insertion pattern but with the split parts matching to different strands or chromosomes.

In RNA-seq, reads come from transcribed sequences with introns and intergenetic regions. When a read is mapped to the genomic reference sequence, it may cross splice-sites, i.e. exon-intron boundaries, and fail to map analogous to a genomic deletion (Li and Homer, 2010). In gene fusions, different genes were combined producing new transcripts (Iyer et al., 2011). RNA-Seq reads covering gene fusions may cross more complex breakpoints from intra- or inter-chromosomal translocations.

Different methods apply different strategies to find split-read matches and usually put constraints on the reference matching locations (see Section 3.1). Split-read approaches are the most promising ones developed so far in the sense that they can be applied to all SV types and are capable of detecting the exact breakpoint at base-pair resolution. This

breakpoint is the split point in the read sequence where the alignment of the read to the reference is broken. However, split-read approaches usually require longer reads and lack power in repeat or duplication regions (Mills et al. (2006), Alkan et al. (2011)). Also, searching for short gapped local alignments is computationally expensive. To reduce this computational overhead, various strategies are used. For example, the Pindel algorithm (Ye et al., 2009) uses paired-end data information to reduce the sequence space that has to be scanned (Alkan et al., 2011).

We explain existing tools and approaches of split-read mapping in the following chapter.

3. Related Work

In this chapter, we review existing split-read approaches for SV and splice-site detection. Computation of precise breakpoints is introduced in Section 3.2.

3.1. Split-read Mapping for SV and Splice-site Detection

Split-read mapping approaches take advantage of reads crossing a breakpoint (see Section 2.3), which enables them to detect the exact breakpoint of a SV or exon-exon junction (Emde et al., 2012).

There are often additional constraints on collinearity, maximum allowed distance between matches, or the location of matches (e.g. Emde et al. (2012)).

The most challenging part in using split-read approaches is to define the split positions of the read. Most methods artificially split the read in two or three parts of equal length, and then use a global alignment approach to find both parts. Using global alignment approaches gives an advantage on the runtime side, since there are fast global read mapper like Bowtie, BWA or RazerS (Li and Durbin (2010), Langmead et al. (2009), Weese et al. (2009)). However, the split can occur at almost any position within the read. Artificial splits may therefore result in either missing the correct split so that the split-read does not map, or in shorter and maybe therefore ambiguous split parts.

One way to cope with this would be to successively split the read in portions of two with different lengths per portion (given a minimal required length for a portion) (Zhang and Wu, 2011). The read then has to be mapped several times with different split parts.

The most recent split-read method that does this is the split-read identification, calibrated (SRiC), by Zhang et al. (2011). It only detects small or middle sized insertions and deletions and analyses the insertion sequence for novel insertions, duplications or translocations. The authors claim that current SV detection methods have multilevel biases in their identifications, and therefore call, e.g., more deletions than insertions. Their key aspect to overcome this issue is to calibrate their method with a data set from the 1000 Genomes project. However, their method is not available for public use so far.

A more natural approach is to use a local alignment approach on the whole read, allowing for flexible split points in the read. Local alignment methods are generally slower than global alignment methods though. We will use Stellar as a local aligner in the first part of our tool (see Section 4.1).

SplazerS is specializing on detecting medium-sized insertions and long deletions by allowing for one split and collinear split parts on one strand (Emde et al., 2012). Prefix

and suffix part do not have to be of a specified length. It also computes the precise breakpoint analogous to Abyzov and Gerstein (2011). It is also novel in that it supports both edit (gapped) and Hamming (ungapped) distance.

Many split-read methods are applied to RNA-Seq. SpliceMap (Au et al., 2010) splits 50 bp reads into 25 bp half-reads, and then uses a standard read mapper like Bowtie, SeqMap, or Eland. For reads longer than 50 bp, they divide the read into overlapping segments of 50 bp and run the same algorithm on the segments. Matching half-reads are connected to determine exon and possible junction locations. Unmapped half-reads are used to locate possible novel junctions of new transcripts. SpliceMap then looks for the canonical GT-AG splice site signal to confirm the junction. The junction has to be supported by a number nR of reads. In the end, they use paired read information (if available) to reduce false discoveries. The method depends on the sequencing depth to support reliable detection of splice junctions that are only present at a low level.

RNA-Seq reads are of particular interest for novel isoform discovery, i.e. unknown RNA transcripts of genes. Therefore, the split-read approach is quite popular here but existing approaches often take additional information of known splicing junctions into account (Li and Homer, 2010).

Karakoc et al. (2011) also use a combined paired-end split-read method to detect SVs and indels within exome data. Their combinatorial algorithm *Splitread* first uses *mrsFast* to locate unmappable paired reads using their anchored partner (one end-anchored placements). mrsFast uses a seed-and-extend algorithm, and finds all occurrences of a read given a predefined Hamming distance. It artificially splits the unmapped orphant reads into subsequences of either equal (*balanced*) or unequal (*unbalanced*) length, and searches for clusters of the split-reads. The method applies a weighted set-cover approximation and uses a maximum parsimony-based general framework to identify the underlying breakpoints of the splits. Although they use the term SV, Splitread only detects indels of different sizes (small indels <50 bp and larger indels ("SVs")).

Zhang and Wu ((Zhang and Wu, 2011)) also support their split-read results by paired-end data information. If the difference of the discordant insert size and the length of the candidate deletion matches the library insert size, it strongly indicates that the candidate is a true deletion.

Besides SplazerS, all existing split-read tools are restricted to either single or paired-end data. It happens so, that all split-read methods allowing for all four SV types, namely insertions, deletions, inversions and translocations, only support paired-end data (Xi et al., 2011). Our method will be the first one supporting single-end reads, and still allowing for all four types of SVs.

So there is actually no state-of-the-art-tool or method based on a split-read approach that is comparable to our method. They are either dependent on paired-end data information or have too many constraints on SV types. Instead, we tested our method on a simulated dataset from Illumina reads. Together with the data, Illumina provided us with the results that their SV detection pipeline Grouper, and the tool SVDetect yielded on this dataset. Both methods are paired-end based, and follow a similar approach.

Illuminas Grouper pipeline uses orphan reads anomalous read pairs for variant detection. First, Grouper computes separate clusters of the orphan reads and anomalous

mapping pairs. Clusters corresponding to the same event are combined, and assembled into contigs. These contigs are mapped back to the reference genome, using positions of associated correct mapping reads to narrow down the search space.

The tool SVDetect (Zeitouni et al., 2010) has a similar approach to Grouper. First it also groups anomalous read pairs into clusters most likely belonging to the same SV. It then identifies groups which share a similar genomic location. Next, SVDetect divides the reference genome into overlapping windows of fixes size, and then uses the previously identified similar group to link pairs of windows. The method thereby predicts large insertions and deletions, inversions, duplications, and inter-chromosomal translocations. To improve SV characterisation, SVDetect also analyses CNVs. SVDetect is limited to genomic paired-end and mate-pair data from Illumina GA and ABI SOLiD platforms.

Due to small variation in the reads/sequences, found breakpoints belonging to the same SV are not necessarily equal down to the exact same bp position. Tolerance in variation of the position is often SV dependent (e.g. deletion vs. insertion?) or data dependent (e.g. standard deviation of library insert size for paired-end data) (Zhang and Wu, 2011).

3.2. Breakpoint Computation

Determining the exact breakpoint is necessary to classify and annotate a SV, and to genotype SVs from newly sequenced genomes (Lam et al., 2010). Also, knowing the precise location at single-nucleotide level of a breakpoint reveals information on the functional impact of the SV (Abyzov and Gerstein, 2011). However, it is not as trivial to determine the exact breakpoint. The classical algorithms Needleman-Wunsch and SmithWaterman (or versions of it) are not suitable here to derive the correct biological solution due to their gap penalty schemes (Abyzov and Gerstein, 2011). Current methods therefore usually use a position variance of, e.g., 10 bp when they compare their results to other methods or known SVs (e.g. Karakoc et al. (2011), The 1000 Genomes Project Consortium (2010)).

Abyzov and Gerstein (2011) describe AGE: Alignment with Gap Excision, their method to determine exact breakpoints by aligning a contig sequence to a previously identified SV region. The goal in exact breakpoint computation is to span the SV and perform a precise local alignment at the flanking sites at the same time (Abyzov and Gerstein, 2011). Given two sequences that cross the breakpoint and flank one site of it, AGE simultaneously aligns both sequences, and introduces a large gap or *jump* in the middle that is not penalised. Age then determines the best joining position of both alignments. This dynamic programming approach guarantees to find the optimal, i.e. highest scoring, alignment. This optimal score may not be found if the gap is introduced between already calculated local alignments since the then trimmed alignments are not guaranteed to still be optimal. AGE is also able to resolve a larger SV that is accompanied by a micro-indel.

We will use a similar method that is implemented in SplazerS. However, the current implementation is not yet able to also resolve small accompanying micro-indels. Still,

SplazerS and Pindel are the only methods with a breakpoint resolution on nucleotide level so far (Zhang et al., 2011).

Conclusion

Most of the existing split-read approaches concentrate on a certain SV type, and then often introduce additional specific information, e.g. splice-site patterns for RNA-Seq. Furthermore, they usually put constraints on collinearity or matching locations of the splits. Besides SplazerS (Emde et al., 2012), all existing split-read tools are restricted to either single or paired-end data, and often only support a certain sequencing platform such as Illumina or 454.

All split-read methods allowing for all four SV types, namely insertions, deletions, inversions and translocations, only support paired-end data (Xi et al., 2011). Our method will be the first one supporting single-end reads but is still extendible to paired-end data. While using a local alignment approach, we allow for all four types of SVs. At the same time, our method is independent from the sequencing platform. By using a similar approach to AGE (Abyzov and Gerstein, 2011) that is implemented in SplazerS, we determine the exact breakpoint location at single-nucleotide resolution.

4. Methods

In the first part of this chapter, we explain the methods used in our approach. In the second part, we introduce our test data and benchmarking methods.

One approach to detect SVs is the split-read approach, which we will use and extend in our method. Our goal is to detect all breakpoints within any read, and thereby detect structural variations as named in Section 2.2. Therefore, we want to design a split-read mapping method that has the following advantages compared to existing tools and approaches: First we want the split points to be flexible in number and location, i.e. we want to allow for one or multiple splits at almost any position in the read. Next, we want to support single and paired-end sequencing data, and do not want to be restricted to either short or long reads.

Following these requirements, we aim to cope with the difficulties of finding and classifying SVs as discussed earlier. To achieve these goals, we designed a method using the C++ library SeqAn that (1) generates local matches of reads using SeqAn Stellar, (2) chains these local matches and localises optimal breakpoint positions, and (3) outputs all multi-split-matches, i.e., chains, that obtain a specified minimal score. As an optional step (4), we call SVs supported by multiple reads. Figure 4.1 shows an overview of the algorithm, and illustrates the main steps using a real example from a gene fusion published by Maher et al. (2009).

A split is a broken alignment of the read to the reference genome (Alkan et al., 2011). In other words, we will find local matches of the read that in total should represent the entire read. That is why we will use the exact local aligner Stellar to find the split parts of a read which we will explain in the following section. The split positions are called breakpoints. We will define the attributes and the computation of breakpoints in Section 4.2.3. The chaining of the local matches and the retrieval of chains with a minimum score are explained in Sections 4.2 and 4.3, respectively.

4.1. Generating Local Matches Using Stellar

Instead of looking for global matches of either the entire read (standard read mapping approach) or of artificial split read portions (existing split read approaches), we look for local matches of the read using Stellar. In short, by using Stellar, we allow for flexible split points by searching for local matches of the read. This makes our approach more sensitive for breakpoint detection. Using Stellar has another advantage. Stellar guaranties to find all matches of a given length and maximal error rate. That means, we will not miss any splits above a specified minimal length.

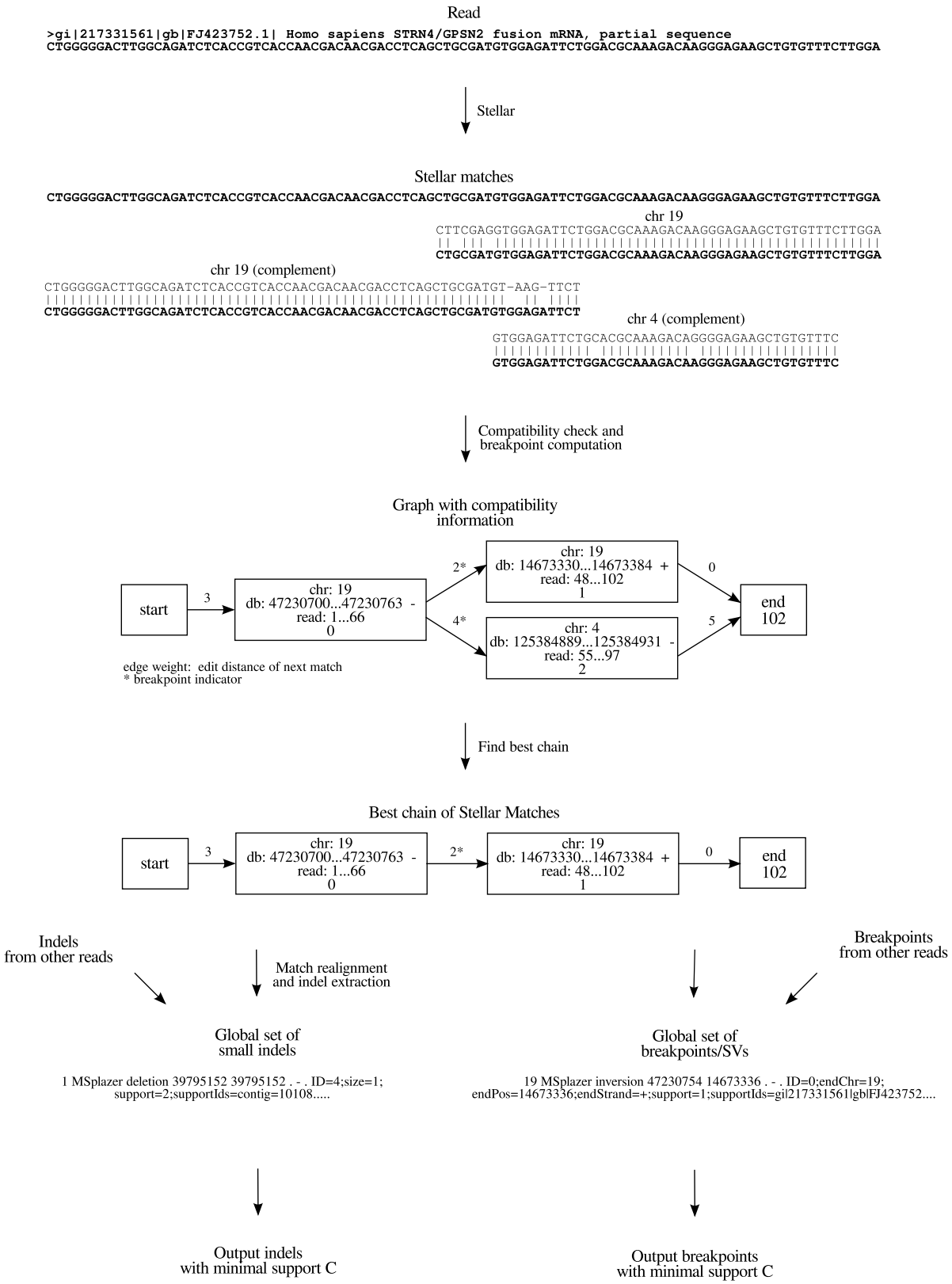


Figure 4.1.: Overview of MSplazer: Illustrated is the outcome of the main steps (1) Stellar match computation, (2) match compatibility check and breakpoint computation, (3) best chain determination via DAG shortest path computation, and (4) breakpoint output. For details on edge score computation see Figure 4.4.

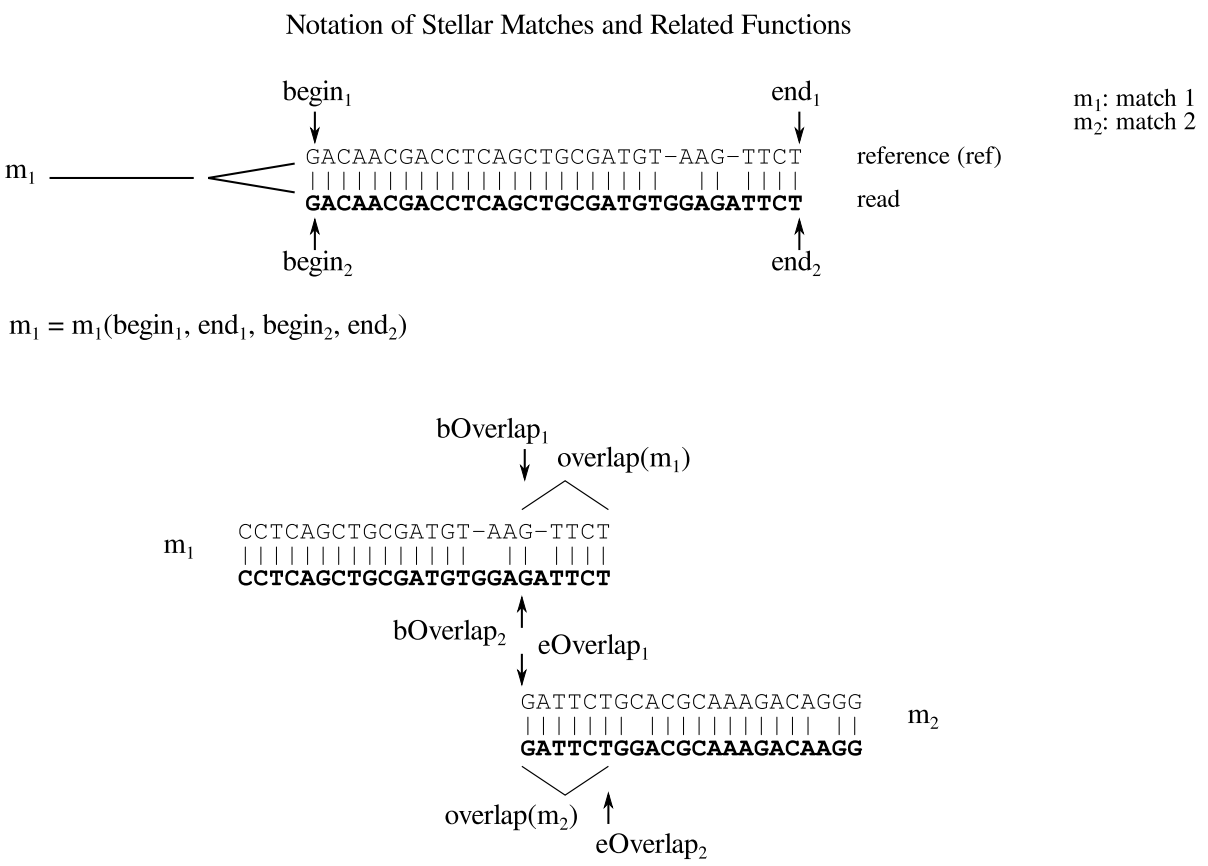


Figure 4.2.: Stellar Match Notation.

Given a maximal allowed error-rate, Stellar uses a seed and extend approach to compute extended exact local matches. A found seed is extended in either direction as long as the errors taken in do not exceed the given error-rate which produces error-prone ends. Due to this extension, two split parts of a read will not only be adjacent but will most likely overlap (see also Figure 4.1). This fact will be used during the chaining of the Stellar matches (see Section 4.2.1).

Our programme reads genome and read sequences from a FASTA file. It supports all Stellar input parameters such as minimum match length, q-gram length and error rate, and calls Stellar for all reads and references. It is also possible to run Stellar externally, and then import the Stellar file with the matches together with the read and reference sequences.

The matches retrieved using Stellar are referred to as *StellarMatches* in the following. StellarMatches belonging to the same read, i.e. the same query, are referred to as *QueryMatches*. In the following sections we will explain how these QueryMatches are chained to represent the whole read. First, we will define some match notations and functions in the following section.

4.1.1. Notation of Stellar Matches and Related Functions

For each read, we have a set of QueryMatches M. We denote a *match* i by $m_i \in M$ with $m_i = (begin_1, end_1, begin_2, end_2)$ where $begin_1, end_1$ and $begin_2, end_2$ are begin and end positions of the alignment in the reference (ref) and the read, respectively (see Figure 4.2). Based on this notation we define the following functions, which we will use throughout this thesis:

$$\begin{aligned} begin(m_1, ref) = begin_1 \quad & end(m_1, ref) = end_1 \quad & begin(m_1) = (begin_1, begin_2) \\ begin(m_1, read) = begin_2 \quad & end(m_1, read) = end_2 \quad & end(m_1) = (end_1, end_2) \end{aligned}$$

$$begin(m_1, read) < begin(m_2, read) \rightarrow m_1 < m_2 \tag{4.1}$$

The equations without a read or reference indicator are used for simplicity when the context or constraint can be applied to both read and reference. Since we will use overlapping matches, we have to define an overlap of two matches m_1, m_2. Begin and end positions of overlaps are denoted with $bOverlap, eOverlap$ (Figure 4.2). We then define the following equations:

$$length(m_1, read) = end_2 - begin_2 \quad length(m_1, ref) = end_1 - begin_1 \tag{4.2}$$

$$length(m_1) = (length(m_1, read), length(m_1, ref)) \tag{4.3}$$

$$overlap(m_1) = m_1(bOverlap_1, end_1, bOverlap_2, end_2) \tag{4.4}$$

$$overlap(m_2) = m_2(begin_1, eOverlap_1, begin_2, eOverlap_2) \tag{4.5}$$

$$overlapFrac(m_i) = length(m_{i'})/length(m_i) \tag{4.6}$$

$$gap(m_1, m_2) = begin(m_2) - end(m_1) \tag{4.7}$$

4.2. Chaining of Local Stellar Matches

Our goal is to gain information about possible split positions within the read which indicate possible SVs. Therefore, we need to combine the matches that belong to the same SV event. In other words, Stellar will probably find quite a few local matches per read, but some of them will stand in conflict to each other or will not cover the whole read when combined. Therefore, we first have to define which local matches are *compatible*. These matches will then be chained, and in the end we will retrieve the chain that we define as most likely.

In the next Section 4.2.1, we will define and explain the criteria we used to determine if two matches are considered compatible. In Section 4.2.2, we will introduce the *split-read graph* representation we used to store all match compatibility information.

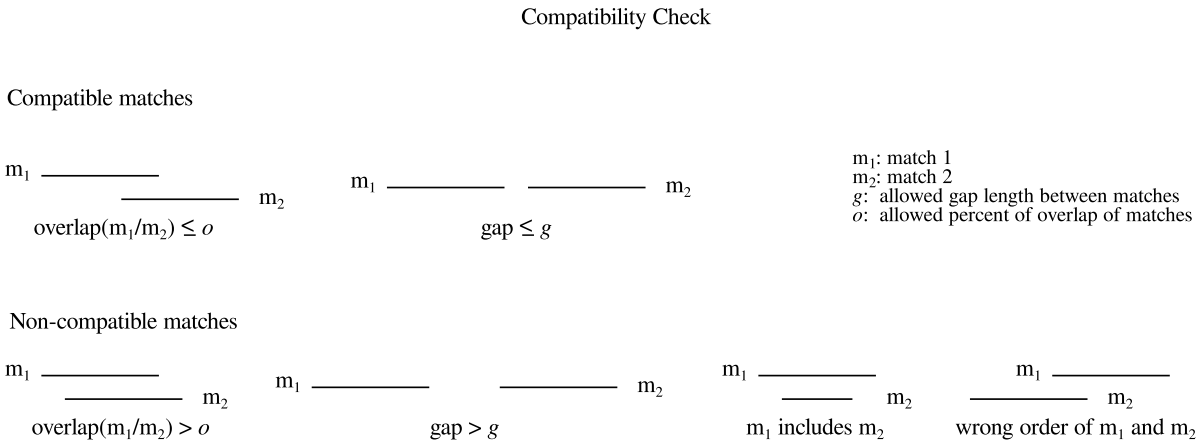


Figure 4.3.: Compatible Stellar matches.

4.2.1. Compatible Stellar Matches

To define the main criterion for compatibility, we make use of one StellarMatch property. In the seed and extend approach of Stellar, an exact mapping seed is extended in one or both directions, as long as the errors taken in do not extend the given error rate. That means, usually a StellarMatch consists of an almost error free seed and, most likely, error prone ends. Due to this property, we expect two adjacent QueryMatches to overlap in either the reference (insertions) or the read (deletions) sequence (see Figure 4.1). We take this as the main criterion of compatibility. If due to a low error rate two matches do not overlap but are adjacent and only have a small gap between them, we will also take these into account as compatible. More details are explained in section Match Overlap and Gap Between Matches.

We divide our criteria in four categories: (1) Match Overlap, (2) Match Distance (Gap Between Matches), (3) Match Location and (4) Match Order. To be compatible, two QueryMatches have to fulfil either the match overlap or the match distance criterion, and then all of the remaining criteria. In the match overlap case, a breakpoint computation (see Section 4.2.3) is done. In the gap case, the breakpoint is set by default to the end position of m_1 and the begin position of m_2. The compatibility information for all QueryMatches is stored in a graph (see Section 4.2.2).

Match Overlap

Two QueryMatches $m_1, m_2 \in M$ can overlap either within the read or the reference sequence. This is our main criterion for compatibility. However, we only want to allow overlaps to a certain point. For instance, it would not make sense to allow an overlap where the overlapping part covers 90% or more of one or both reads. The same thing applies if, e.g., m_2 is fully contained in m_1. In other words, both matches have to fulfil

the following conditions:

$$begin(m_1) < begin(m_2) \ \wedge \ end(m_1) < end(m_2) \ \wedge \ begin(m_2) < end(m_1) \tag{4.8}$$

$$overlap(m_1) > o \ \wedge \ overlap(m_2) > o \tag{4.9}$$

with *overlap threshold* o.

If the SV is a combination of deletion and insertion (Abyzov and Gerstein, 2011), the matches might not overlap. This problem can be coped by allowing a certain gap or distance between match m_1 and m_2.

Gap Between Matches

In some cases, two QueryMatches m_1 and m_2 do not overlap but still belong together, i.e., result from one particular SV. For example, there are cases where a deletion or insertion is accompanied by smaller insertion or deletion, respectively (Abyzov and Gerstein, 2011). In these cases, it makes sense to allow a specified gap of maximal length between m_1 and m_2. Then m_1, m_2 must fulfil the condition:

$$gap(m_1, m_2) < g \tag{4.10}$$

where g is the allowed gap length that can again be specified by an input parameter. The gap is penalised by its length. Note that since the QueryMatches are sorted according to their $begin_2$ positions (see section Match Order), once we found a pair m_1, m_2 which is violating the gap condition, all match pairs m_1, m_i with $i > 2$ will violate it, too, and do not need to be checked any more.

Match Order

The QueryMatches can be ordered according to their positions within the read or the genome.

Order in Read Our goal is to reconnect the QueryMatches in the right order to restore the read. Hence, for two matches m_1 and m_2 with $m_1 < m_2$, it makes only sense to combine them in this order, i.e. chaining m_1 with a directed edge to m_2, and not vice versa. For practical reasons, the matches are therefore sorted, prior to any other comparison, according to their begin position within the read. Thereby we ensure match order compatibility and also reduce the number of compatibility comparisons by only considering matches with a higher order for these comparisons.

Order in Genome We also want to allow for intra-chromosomal translocations. Therefore, we allow arbitrary orders of m_1 and m_2 within the reference sequence. Nevertheless, we introduce a penalty for this event ($different\ order\ penalty$) which can also be specified by the user. This penalty only applies if none of the match location penalties apply (see section Match Location).

Match Location

A SV event within one sequence is more likely to occur than one spanning over different strands (e.g. inversions or gene fusions) or chromosomes (e.g. gene fusions). We will allow for these SVs by not making constraints on the reference sequences of the matches but again introduce a penalty (*inversion penalty*, *translocation penalty*).

4.2.2. Graph Representing Compatibility Between Matches

All QueryMatch compatibility information is stored in a split-read graph representation of the matches. We define the graph $G(V, E)$ as a set of nodes V and a set of edges E where v_i corresponds to the QueryMatch $m_i \in M$ (graph definitions taken from Cormen (2001)). In addition, we have two nodes v_{start}, v_{end} representing start and end of the read. There is a directed edge $e = (v_i, v_j) \in E$ from $v_i \rightarrow v_j$ if the matches m_i, m_j are compatible. There are also edges $e = (v_{start}, v_i)$ and $e = (v_i, v_{end})$ $\forall i$. Each path $P = (v_{start}, ..., v_k, ..., v_{end})$ from v_{start} to v_{end} representing a chain of matches $m_i \in M'$ where $M' \subseteq M$.

In other words, each match is represented by a vertex. Compatible matches are connected by a directed edge which will define a directed acyclic graph (DAG). The graph contains artificial start and end vertices, and each path from start to end represents a possible multi-split-match or *chain* of the read. All chains with a minimal score (i.e. minimal edit distance) can be obtained through graph algorithms.

As a criterion for good matches, we assign to each match its edit distance score, i.e., the number of errors (indels and mismatches) within the match. The edit distance of a match is stored as the weight of the incoming edge of the vertex representing the match (see Figure 4.4). We also add penalties and the breakpoint score (see Section 4.2.3) to the weight. The weight $w(e)$ of $e = (v_i, v_j)$ is then defined as $w(e) = score(v_j) + breakpointScore(v_i, v_j) + penalties$. Figure 4.4 shows the graph, scoring and breakpoint computation of the example shown in Figure 4.1. The vertices contain the match positions within read and database, and indicate the matching strand (+/-).

4.2.3. Breakpoint Computation

Since StellarMatches are extended exact local matches, a correct match has most likely been extended for several bases (as long as the score is still good enough), taking in a few additional errors at the end. If this applies for both matches, it is most likely that, e.g., m_1 has been extended with errors into an area where m_2 already matches but without or with less errors. That means, m_1 and m_2 overlap, but if we cut m_1 at the overlapping end (getting rid of the errors) and take m_2, we have a non repetitive pair, and in addition gained a better score. The cutting point of m_1 and m_2 will be the *breakpoint* of the underlying SV.

During breakpoint computation, the overlapping parts $m_{1'}, m_{2'}$ of two compatible QueryMatches m_1, m_2 are refined. This is done by an alignment refinement implemented

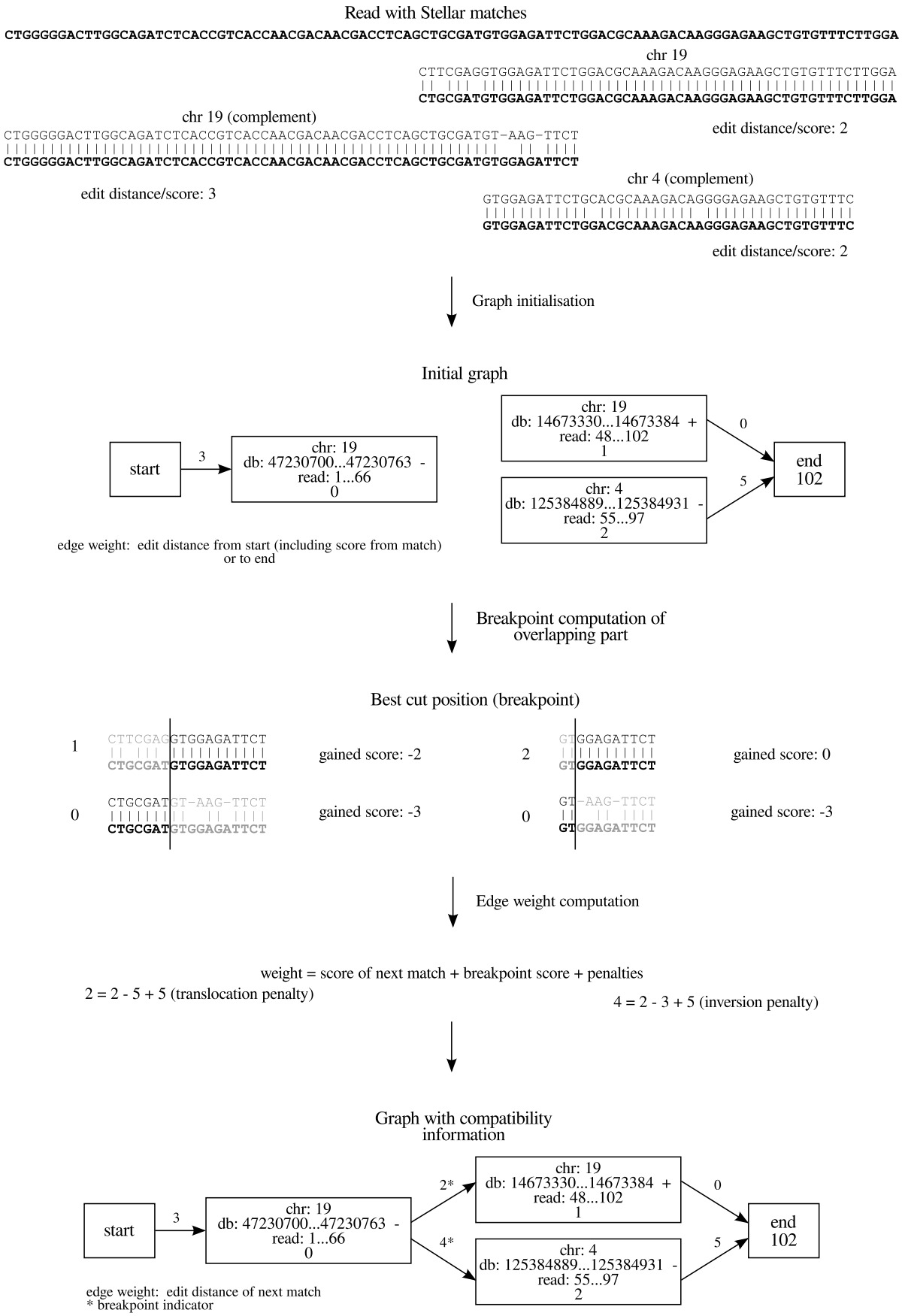


Figure 4.4.: Scoring and breakpoint computation.

in SeqAn's SplazerS (Emde et al., 2012), which is very close to the one explained in Abyzov and Gerstein (2011). The result is a split position that determines the refined end and begin of m_1 and m_2, respectively, and the difference in edit distance gained by the cut. Due to this gain, the edge weight in the graph representation can become negative. This is because we store only the edit distance of m_2 on this particular edge, but the gain can also result from avoiding mismatches or indels in m_1. Since our graph is a DAG, this will be no problem when using shortest paths algorithms later on.

We define our breakpoint $bp(m_i, m_j)$ by $bp = (ref_i, ref_j, orientation_i, orientation_j, refPos_i, refPos_j, SV-type)$. We also keep track of the support of the breakpoint (i.e. the number of reads supporting the breakpoint) and their corresponding read Ids.

4.3. Retrieving Best Chains

Within the scope of this thesis, we only obtain the best chain of a read. To determine the best chain, we use a shortest path algorithm on our split-read graph $G = (V, E)$. Since we have a directed acyclic graph (DAG), we can use the DAG shortest path algorithm, which is already implemented in SeqAn. Shortest paths are well defined in a DAG, and we can compute the shortest path in $\Theta(V + E)$ time (Cormen, 2001).

Multiple Read Support

The program Pindel (Ye et al., 2009) used a cutoff value C to determine deletions. Pindel version $v\_0.1.0$ uses a default value of $C = 2$, version $v\_0.2.0$ a default value $C = 3$. SVseq (Zhang and Wu, 2011) uses the same value C as a threshold for minimal multiple read support in their split-read method. We will use this value as well as a default value for our multiple read support but allow it to be specified as an input parameter.

4.4. Evaluation Data

We use three different datasets to validate our method including RNA-Seq as well as DNA-Seq data. With these three sets we cover Illumina and 454 read data from simulated and real data sets. In short, the sets contain:

1. Sequences of chimeric transcripts
2. Simulated data from Illumina read data
3. Real data from 454 RNA-Seq data

We will explain the three datasets in detail in the following sections.

Sequences of Chimeric Transcripts

The first set consists of 14 sequences of chimeric transcripts published in Maher et al. (2009) (Supplementary Table 9). The size of the sequences varies between 92 bp and

256 bp (see table 4.4). The junction positions in table 4.4 indicate the split sizes of the sequences. The set includes breakpoints from inversions, deletions and inter-chromosomal translocations, and two sequences map partly to high repeat regions.

Gene transcript & GenBank AN	Sequence length	Fusion junction position
INPP4A-HJURP FJ423742	245	158
ZNF649-ZNF577 FJ423743	229	169
TMPRSS2-ERG FJ423744	235	80
USP10-ZDHHC7 FJ423745	232	82
HJURP-EIF4E2 FJ423746	244	97
MIPOL1-DGKB FJ423747	256	149
MRPS10-HPR FJ423748	149	38
WDR55-DND1 FJ423749	255	178
MBTPS2-YY2 FJ423751	233	105
STRN4-GPSN2 FJ423752	101	55
LMAN2-AP3S1 FJ423753	99	59
RC3H2-RGS3 FJ423754	101	51
SLC45A3-ELK4 FJ423755	92	50
C19orf25-APC2 (Intron) FJ423750	249	187

Simulated Assembled Contigs from Illumina Read Data

The second set includes simulated data from Illumina reads (not published). They are contigs produced by Illuminas Grouper pipeline. To generate the set, first a set of SVs, including indels, inversions and duplications, was integrated into the human chromosome 10 (hg18). We will later use this set as a reference set to validate our results. Paired-end reads from this altered chromosome were simulated and mapped to chromosome 10 (hg18). Grouper first clusters anomalous or bad mapping reads, and then assembles contigs from theses clusters. Illumina provided us with a set of $103,101$ contigs from this simulation, consisting of good and also false contigs. The contig length ranges from 37 bp to 607 bp, and the average contig length of the set is 152 bp.

Real Data (RNA-Seq 454 reads)

The third set consists of 454 RNA-Seq single-end reads from human whole-transcriptome sequencing of melanoma metastases (Feldhahn et al., 2011). The set has $92,296$ reads varying from 100 bp to 585 bp in size. We chose this additional dataset because multiple, i.e. more than one, splits are more likely to occur in RNA-Seq data than from SVs in genome sequencing data, and we wanted to test how our method performs on RNA-Seq data.

Parameter	default value	lower bound	upper bound
overlap (o) *in* %	50	0	100
gap (g) *in* bp	10	0	1,000
initGap in bp	15	0	1,000

Table 4.1.: Test values for main parameters.

4.5. Evaluation Methods

First, we do a parameter test for our main parameters, i.e. the allowed overlap o of compatible matches, the allowed gap length g between matches (see Figure 4.3 for both), and the length of an initial or end gap *initGap*. Table 4.1 shows the default and test values for the parameters. When the overlap parameter is set to 0, predictions result only from the allowed gap between matches. We conduct this parameter test with the Illumina contig data (Section 4.4).

As a proof of concept and qualitative analysis, we use the 14 sequences of chimeric transcripts (see Section 4.4) as read input. Our method should be able to detect all splits and their junctions. Next, we analyse the results for the Illumina contig data.

SV Detection in Contigs from Simulated Illumina Data Illumina provided us with a reference set which includes all the SVs they inserted into their simulated data. We compare our predictions with this reference set, and then analyse the results for sensitivity and precision by determining recall and positive prediction values (PPV). We use the equations

$$sensitivity(Recall) = \frac{TP}{|SV_{ref}|} \tag{4.11}$$

$$precision(PPV) = \frac{TP}{|SV_{pred}|} \tag{4.12}$$

where TP is the number of true positive predictions.

For the comparison, we extend and apply a method used for indel comparison in SplazerS (Emde et al., 2012). Here the authors adopted the computation of the *extended indel region* (Krawitz et al., 2010), which accounts for repeats, and allows a window for the location of the indel. The method also allows a variation of the indel size. We extended this method to also recognize and compare inversions and duplications.

As default values, we use a window of 5 bp for the SV position and allow a variation of 20% for its size (if applicable). We compare our results to those from Illuminas Grouper pipeline and the tool SVDetect. All results from both tools were kindly provided to us by Illumina.

RNA-Seq Data Analysis of 454 Single-end Reads For the RNA-Seq data, we built a reference set that consists of all possible exon-exon junctions derived from all RefSeq

genes (Pruitt et al., 2009). We verify all the deletions we predicted against this set since they should correspond mostly to introns.

The second set includes SV predictions from the 454 analysis pipeline on the same data (Roche Diagnostics Corporation, 2012). We compare the results to those yielded by the 454 analysis software which were provided to us along with the dataset. The 454 analysis pipeline uses an assembly approach of paired-end read data, similar to Grouper.

We apply the specificity and sensitivity analysis from the previous section to evaluate the predictions. We use a window of 1 bp for the SV position and allow a variation of 5% for deletion size as default setting.

Summary

We implemented a new split-read approach within the software library SeqAn that uses SeqAn's Stellar. The main steps are (see Figure 4.1 for detailed overview):

1. Initialization
    a) Read genomes and reads from FASTA files
    b) Run Stellar (support of all Stellar options) or import Stellar matches (if file is available to program)
2. Chaining Stellar matches
    a) For each read: Initialise graph structure and insert edges between compatible vertices(matches)
3. Find Best chain(s)
    a) Find best chain via shortest path computation
    b) Get breakpoints from edges and matches
4. Breakpoint Analysis
    a) Report breakpoints with minimum support
5. Output
    a) Breakpoint out file
    b) Chain out file in SAM format
    c) Graph output in dot format

We will evaluate the method on three datasets, including Illumina and 454 reads from simulated and real data. Results and evaluation will be discussed in the next chapter.

5. Results and Discussion

In this chapter we will present the results to the tests described in chapter 4.5. We will start with the parameter and runtime analysis, and then evaluate the data from the chimeric transcript sequences, the simulated Illumina contigs, and the 454 RNA-Seq reads. All benchmarks were performed on a machine with 2 x Intel(R) Xeon(R) CPU X5550 @ 2.67GHz and 50GB of RAM.

5.1. Parameter and Runtime Analysis

We conducted the computation of Stellar externally, and used read files of $5,000$ reads to run in parallel to reduce the runtime. The setting for Stellar is $error\ rate = 0.03$, $min.\ length = 30$ and $kmer = 16$ (where different from default values). Each Stellar run with $5,000$ reads and the whole human genome as reference took approximately 20 hours for the Illumina contigs and eight hours for the 454 RNA-Seq reads. The MSplazerS runtime for the Illumina contigs was 41 min, most of it being I/O runtime to import the genome, reads and the $1,802,417$ Stellar matches (37 min I/O). The MSplazerS runtime for the 454 RNA-Seq reads was 25 min (21 min I/O) I/O with $1,125,998$ Stellar matches). So the bottle neck for our method is computing the local matches, as we expected. To improve runtime, one approach could be to enhance Stellar to process all the reads parallel. The computation of matches for one read is independent of the other reads.

We tested upper and lower bound values for the main parameters of the programme (see table 4.1) using the Illumina contig dataset. To evaluate the tests, we look at the number of predictions for each value. Results are shown in table 5.1. Only the breakpoints from the edges are shown since the parameters do not have an effect on the indels from the matches.

For the default values $overlap(o) = 0.5$, $gap(g) = 10$ bp and $initGap = 15$ bp, our method finds 17,382 predictions. When we set the overlap to $o = 0$, we still see 8,606 predictions, which is more than we had expected. As an additional sanity check, we set both $o = 0$ and $g = 0$, to see if all the predictions were really from the small gaps. As a

Parameter	default value	lower bound	upper bound
overlap (o) *in*%	17,382	8,606	18,424
gap (g) *in* bp	17,382	17311	17,730
initGap in bp	17,382	413	17,717

Table 5.1.: Number of predicted breakpoints for given parameter setting.

result, we yield only three predictions. This shows, that the overlap parameter is really the main parameter which supports our approach of using the overlap of Stellar matches as the main criterion. But there seem to be a lot ambiguous matches resulting from a small gap between the matches. The gap parameter itself has only a little influence on the number of predictions.

Another large drop results when setting the initial gap parameter to $initGap = 0$ (drop to 2.5%). This drop is a bit unexpected. Leading and trailing gaps seem to be an issue in the contigs but the gap value of 10 bp is sufficient to overcome these gaps. The remaining numbers only increase to a small amount. In the next section we will shortly analyse the general sensitivity of our method.

5.2. Graph and Chain Size Analysis

To give an idea of the general sensitivity, we will again use the Illumina contig dataset to perform some statistic analysis on the number of matches and splits per contig. First, we will look at the total number of contigs covered by a chain and their chain size distribution. For 80% of the 103,101 contigs we found a complete chain (Figure 5.1). For another 17% we could determine a partial chain, where either the first or the last part of the contig is covered by a local match (note that these partial chains are not yet evaluated by the method). That means, only for 3% of the contigs we could either not find any local matches or the matches found were not compatible. The chain size distribution for the complete chains is shown in a bar plot in Figure 5.2. The longest chain we found has five matches. 99.5% of the complete chains have either one match (no split) or two matches (one split), but we do see chains with 3, 4 or 5 matches. Since we expect to see more splits when using the 454 RNA-Seq reads, we included them for comparison in the chain size distribution in Figure 5.2. Here we have chains up to 9 matches in length, and most of the chains have up to 5 matches (note log scale in Figure 5.2).

Next, we will look at the distribution of the graph size, i.e. the number of local matches found for each contig, and the corresponding number of complete and partial chains. We have a total of 103,101 contigs, 54% of which have either one or two matches. Most of the remaining contigs have up to 50 matches, with 30,906 contigs (30%) having exactly 50 matches. Stellar has an abundance cut parameter which is set to 50 matches. That means, all these 30,906 contigs have either 50 or more matches which indicates that the contigs are lying within repeat regions. Values higher than 50 matches arise from contigs with a lot of inverse matches (Stellar's abundance cut parameter only takes one strand into account). Figure 5.3 shows the ratio of (1) contigs with complete chains, (2) partial chains, and (3) no chain to the total number of contigs, distinguished by the number of local matches per contig. The ratio does not change much with increasing number of matches which indicates that we find complete chains independent from the initial match number, and also even for matches within high repeat regions. The accuracy of the predictions arising from these chains will be evaluated in Section 5.4.

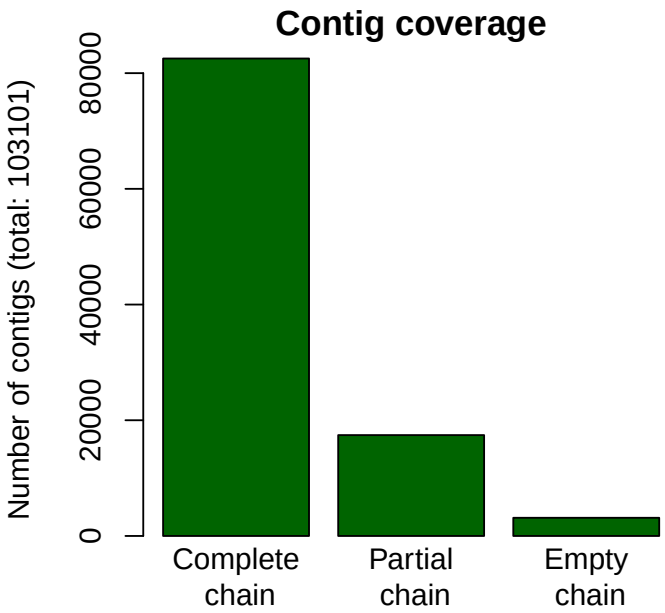


Figure 5.1.: Contig coverage: Relation of total number of complete, partial (only begin or end of contig found) and empty chains (no matches found).

5.3. Proof of Concept and Qualitative Analysis Using Sequences of Chimeric Transcripts

First, we tested our method with 14 published sequences from chimeric transcripts (see Section 4.4). For Stellar we used $error\ rate = 0.05$ and $min.\ length = 30$. Since we only have one sequence per breakpoint instead of multiple reads, we set the required support to $support = 1$. The predicted breakpoints are shown in table 5.2. For all 14 sequences, we found the correct splits as the best chain. For sequences C19orf25-APC2 (Intron) FJ423750 and MRPS10-HPR FJ423748, we found a second split belonging to an exon-exon boundary. These two examples show that our method is able to find multiple splits per read at variable positions. The breakpoints we found include inversions, deletions on both strands, and inter-chromosomal translocations.

Compatibility graphs including the best chain are shown in Figure 4.1 and in appendix A.1. For 8 of the 14 sequences, there are only two matches and one chain. For another 5 sequences, Stellar found several ambiguous matches resulting in up to 10 alternative chains per graph. For sequence C19orf25-APC2 (Intron) FJ423750, Stellar found 47 nearly identical matches for the last split (see Figure A.15). For comparison, we reran the last sequence using $min.\ length = 50$ (see Figure A.14) where the number of ambiguous matches was reduced to 5. These last 6 examples show that our method can also handle a great number of ambiguous matches from repeat regions. However, as seen in Figure A.7, there are cases where we have multiple best chains per read. In the current version, our method arbitrarily chooses one of them.

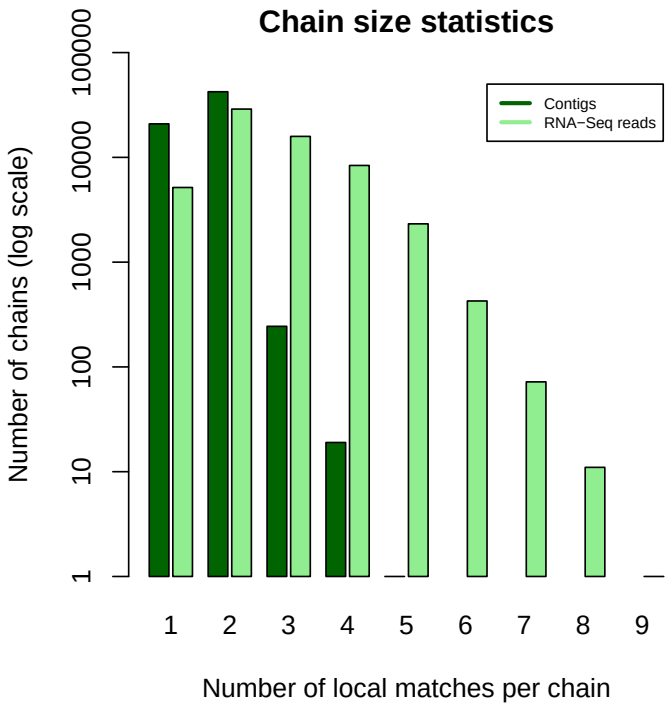


Figure 5.2.: Chain size distribution: Number of chains with 1 to 9 (observed maximum) matches per chain for Illumina contigs and 454 RNA-Seq reads. Note log scale of the number of chains.

5.4. SV Detection in Contigs From Simulated Illumina Data

Next, we tested our method using a simulated dataset containing contigs from Illumina reads (see Section 4.4). The setting for Stellar is $error\ rate = 0.03$, $min.\ length = 30$ and $k-mer\ length = 16$ as for the parameter test performed on this dataset. Since the contigs already arise from multiple reads, we will set the support parameter for the breakpoints to $support = 1$. MSplazer overlap and gap parameters are set to default values, penalties are $translocation\ penalty = 5$, $inversion\ penalty = 5$, and $order\ penalty = 0$ (default penalties). As described in the previous chapter (4.5), we will compare our predictions with a reference set including the inserted SVs using the variances of 5 bp for the positions and 20% for the variant size. All SVs from the reference set are on chromosome 10 only. That means, all inter-chromosomal translocation predictions are per sefalse positives, and are excluded by the variance comparison tool and therefore also in the following result tables. We will compare our results with the results from Illuminas Grouper SV analysis pipeline and the results from SVDetect, both provided to us by Illumina.

Table 5.3 shows the total number of SVs in the reference set (SVs_{ref}), the predicted SVs (SVs_{pred}), and the total numbers of true positives (TP), false positives (FP), and the number of false negatives (FN). Note that we considered only chains with

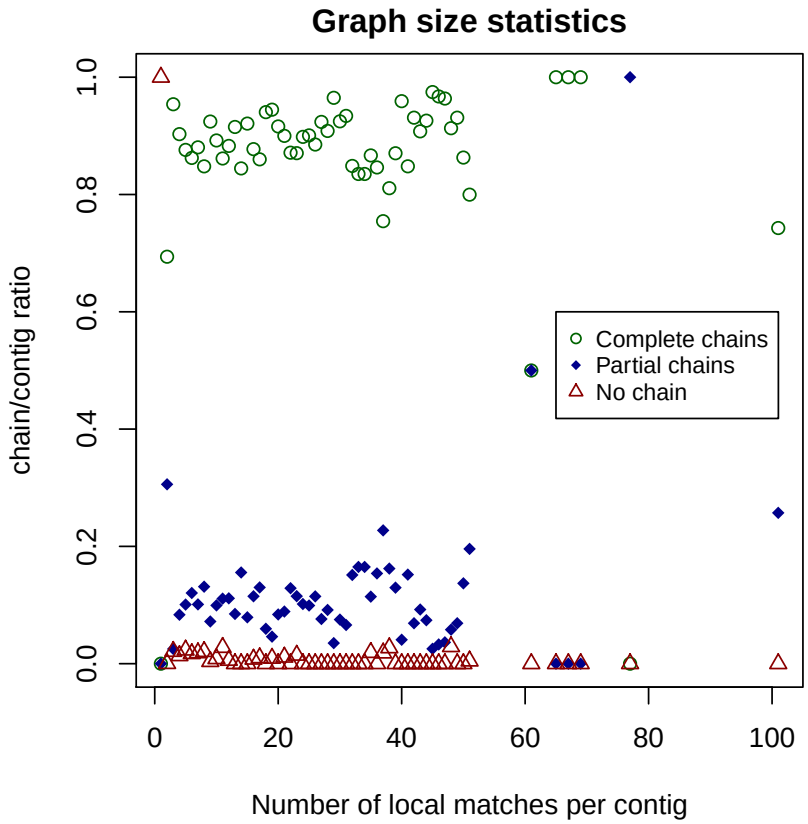


Figure 5.3.: Ratio of (1) complete chains (circles), (2) partial chains (diamonds), and (3) no chain (triangles) to the total number of contigs distinguished by the number of local matches per contig.

at least two matches, i.e., at least one real split and breakpoint, for the results. The MSplazer (large SVs) results in table 5.3 are predictions only from breakpoints between split parts (edges in the compatibility graph). The MSplazer+S results also include the smaller indels taken directly from the matches. Regarding the large SVs, the MSplazer approach yielded a total of 16,925 SV calls, 16,026 (94.7%) of which were included in the reference set. However, we only covered 54.7% of the reference set. We found that the missing SVs are mainly small indels <10 bp (see tables 5.4 and 5.5), and therefore took the indels from the matches into account.

When we include the small indels from the matches, our method yields additional 10,707 (60%) predictions. So, the sensitivity increases to 63.3%, that means we actually do find more of the smaller indels we missed without the matches. Unfortunately, we also gain a lot more false positives so that the specificity drops to 66%. We still do retrieve indels from the matches but compare them separately (i.e. a set with and a set without the Stellar indels as above) to the results from Grouper and SVdetect on this dataset.

GenBank AN	Pos	Breakpoint
FJ423742	158	chr2 inversion 234,746,299 (+) 99,193,526 (−)
FJ423743	168	chr19 deletion 52,384,129 (+) 52,394,083 (+)
FJ423744	77	chr21 deletion 39,817,397 (−) 42,880,073 (−)
FJ423745	79	chr16 inversion 84,733,715 (+) 85,024,100 (−)
FJ423746	97	chr2 inversion 233,421,216 (−) 234,749,254 (+)
FJ423747	149	chr14 translocation 37,969,344 (+) (chr7) 14,188,763 (+)
FJ423748	38	chr16 translocation 72,094,042 (−) (chr6) 42,176,037 (+)
	(112)	chr6 deletion 42,176,111 (+) 42,176,595 (+) (intron)
FJ423749	175	chr5 deletion 140,049,273 (+) 140,050,850 (+)
FJ423751	102	chrX deletion 21,871,619 (+) 21,874,597 (+)
FJ423752	55	chr19 inversion 47,230,754 (−) 14,673,336 (+)
FJ423753	59	chr5 inversion 176,778,501 (−) 115,202,364 (+)
FJ423754	51	chr9 inversion 125,627,674 (−) 116,299,075 (+)
FJ423755	50	chr1 deletion 205,593,020 (+) 205,630,988 (+)
FJ423750	185	chr19 deletion 1,461,766 (−) 1,478,910 (−)
	(50)	chr19 deletion 1,478,778 (−) 1,479,204 (−) (intron)

Table 5.2.: Breakpoints for the chimeric transcript sequences: Breakpoints include inversions, deletions, and inter-chromosomal translocations. Matching strands are indicated by (+/−). Pos = predicted junction position on read.

	MSplazer (large SVs)	MSplazer+S (incl. small indels)
$\|SV_{ref}\|$	29,771	29,771
$\|SV_{pred}\|$	16,925	27,632
TP	16,026	18,224
FP	899	9,408
FN	13,498	10,915

Table 5.3.: Total number reference SVs ($|SVs_{ref}|$) and predicted SVs ($|SVs_{pred}|$, excluding inter-chromosomal translocations), and the number of true positive predictions (TP), false positives (FP), and false negative predictions (FN).

Comparison to Grouper and SVDetect

We will compare our results to Grouper and SVDetect classified by SV type, i.e. we distinguish deletions, insertions, inversions and duplications, and divided into different size ranges (1 to 299, 300 to 10, 000 and $> 10,000$). Tables 5.4 to 5.7 show the results in terms of sensitivity (Recall) and specificity (PPV) for all methods. Criteria of Grouper and SVDetect for valid calls: SVDetect requires 90%

Deletions For deletions <300 bp, the Grouper pipeline has by far the highest sensitivity (91.5%), and at the same time a high specificity (PPV) (96.8%). MSplazers specificity is similarly high (95.7%) but with a low sensitivity (53.1%). For MSplazer+S the sensitivity increases to 75.4% but at the price of a much lower specificity (46.2%). SVDectect did not detect any deletions here. As mentioned above, the low sensitivity of MSplazer is due to the fact that MSplazer (without the Stellar indels) is not necessarily made to detect small indels <10 bp but larger SVs. For deletions from 300 to 10,000 bp, MSplazer has the highest specificity (PPV) (95.6%) with a moderate sensitivity (74.1%). Grouper has again the highest sensitivity (84.4%) but with a much lower specificity 65.1%. There were only 3 deletions >10,000 bp in the reference set. MSplazer and SVDetect called all of them, Grouper none (but can localise them using their so called *open-ended*, i.e. partially mapping, contigs). Otherwise, for large deletions, Groupers current implementation is limited to 10,000 bp. SVDetect also has a high PPV (80%) where MSplazer calls a lot more wrong deletions (PPV 0.03%).

Insertions As for deletions, Grouper has the highest sensitivity (88.1%) and specificity (99.2%) for insertions <300 bp. MSplazers specificity (PPV) is high (94.4%) but again with a low sensitivity (54.1%). MSplazer+S sensitivity is much closer to Grouper this time (75.9%), and the specificity does not drop that much (76.9%). There are again no calls from SVDetect for this size range. MSplazer and Grouper called no events for size range 300 to 10, 000, only SVDetect called 5 false positive insertions here. SVDetect only detects large insertions and deletions within 2-3 SD from the mean insert size of the paired-end library, and is therefore limited to this size range. For large insertions, Grouper, too, is limited by the read and insert size, so it can only get the boundaries, but not the core. The read size limitation also holds for MSplazer. It would be possible to call the insertion boundaries from partial chains, but the partial chains are basically unsupported local matches and therefore much less reliable than complete chains. Also, it is hard to determine the exact breakpoint from a boundary match. These problems are assessed by specialised insert assembly methods.

Inversions For inversions, we do not distinguish between size ranges. MSplazer is by far the most sensitive method (80.2%), and similarly specific as SVDetect (97.4%, 98.2%). Both Grouper and SVDetect need special anomalous read information to correctly identify inversion clusters, and also need the complete read or contig to map to the reference. Here, MSplazer can profit from the generic split read approach.

	MSplazer		MSplazer+S		Grouper		SVDetect	
$Deletion_{ref}$	Recall	PPV	Recall	PPV	Recall	PPV	Recall	PPV
1 to 299 (10598)	53.1	95.7	75.4	46.2	91.5	96.8	0	0
300 to 10000 (767)	74.1	95.6	74.1	95.6	84.4	65.1	53.7	62.0
>10000 (3)	100	0.03	100	0.03	0	0	100	80

Table 5.4.: Number of reference deletions ($|Del_{ref}|$) per size, and sensitivity (Recall) and specificity (PPV) values for MSplazer, MSplazer+S (i.e. including small indels), Grouper and SVDetect.

	MSplazer		MSplazer+S		Grouper		SVDetect	
$Insertion_{ref}$	Recall	PPV	Recall	PPV	Recall	PPV	Recall	PPV
1 to 299 (10515)	54.1	94.4	75.9	76.9	88.1	99.2	0	0
300 to 10000 (837)	0	0	0	0	0	0	0	0 (5 FP)

Table 5.5.: Number of reference insertions ($|Insertion_{ref}|$) per size, and sensitivity (Recall) and specificity (PPV) values for MSplazer, MSplazer+S (i.e. including small indels), Grouper and SVDetect. There are no insertions >10,000 bp in the reference set, and neither method called any false positive events.

	MSplazer		Grouper		SVDetect	
$Inversion_{ref}$	Recall	PPV	Recall	PPV	Recall	PPV
(1119)	80.2	97.4	41.7	41.1	66.6	98.2

Table 5.6.: Total number of reference inversions ($|Inversion_{ref}|$), and sensitivity (Recall) and specificity (PPV) values for MSplazer, Grouper and SVDetect.

Duplications MSplazer does not explicitly call duplications but we regarded all intra-chromosomal translocation predictions within distance of 500 bp as possible duplications. The reference set does not include duplications <300 bp, and neither Grouper or SVDetect predicted any duplications of this size range. MSplazer has 13 false positive predictions. Duplications or translocations in general within this size range are indeed very unlikely, and are most likely wrong classified small indels. For duplications from 300 to 10,000 bp and >10,000 bp, SVDetect is the most sensitive (88.0%, 90.6%) and specific (99.4%, 99.8%) tool. MSplazer and Grouper are equally specific for 300 to 10,000 bp, MSplazer also for >10,000 bp, and also by maintaining a high sensitivity (80.7%, 78.5%). As for inversions, Grouper needs special anomalous read information to correctly cluster the contigs. SVDetect uses a complementary analysis of CNV to improve SV characterisation. Especially duplication identification benefits from this additional functionality.

5.5. RNA-Seq Data Analysis of 454 Single-end Reads

Finally, we tested our method on a real dataset containing 454 RNA-Seq single-end reads (4.4). The setting for Stellar is $error\ rate = 0.05$ and $min.length = 30$. The penalties for MSplazer are $translocation\ penalty(tp) = 15$, $inversion\ penalty(ip) = 15$, and $order\ penalty(op) = 10$ (default is $tp = 5, ip = 5, op = 0$). The penalties are set higher than for the contigs since we expect mainly deletions resulting from introns in RNA-Seq data and therefore want to favour them over more complex SVs. The increase of 10 for all penalties was chosen relatively arbitrary, a careful investigation remains as future work.

We used $support = 2$ and $support = 3$, and will compare both results to the results from the 454 analysis pipeline. First, we will compare all results to the exon-exon junction reference set, and use the percentage of deletions confirmed by introns as an indicator for precision (PPV). For comparison to the reference set, we use two variance settings. The default setting is 1 bp for position variance and 5% for the deletion size (*pt1st5*), the second allows no variance at all (*pt0st0*). In addition, we will look at the overlap of predictions from MSplazer and the 454 analysis pipeline.

Table 5.8 shows the total number of deletion predictions as well as true positives and false positives values for MSplazer with support=2 and support=3 and for the 454 pipeline, given the default variance setting *pt1st5*. The 454 analysis pipeline yielded 3,774 predictions, 3,524 (93.4%) of which were confirmed by the reference set. MSplazer with support=2 yielded 6,032 predictions, many more than the 454 pipeline, 5,080 (84.2%) of which are true positive exon-exon junctions. For support=3, MSplazer finds 1,958 predictions, where 1,710 (87.3%) are true positives.

Table 5.9 shows the same predictions compared to the reference set allowing no variance at all (*pt0st0*). MSplazer now yields 4,660 (77.3%) true positives for support=2, and 1,621 (82.8%) for support=3. The 454 pipeline yields 2,685 (71.1%). So, even when we require the predictions to be exact in size and position, both methods show high precision in their predictions.

	MSplazer		Grouper		SVDetect	
$Duplication_{ref}$	Recall	PPV	Recall	PPV	Recall	PPV
1 to 299 (0)	-	0 (13 FP)	-	0 (0 FP)	-	0 (0 FP)
300 to 10000 (550)	80.7	95.1	44.0	93.2	88.0	99.4
>10000 (576)	78.5	93.7	1.0	0.5	90.6	99.8

Table 5.7.: Number reference duplications ($|Duplication_{ref}|$) per size, and sensitivity (Recall) and specificity (PPV) values for MSplazer, Grouper and SVDetect.

pt1st5	MSplazer support=2	MSplazer support=3	454 pipeline
Predictions	6,032	1,958	3,774
TP	5,080	1,710	3,524
FP	952	248	250

Table 5.8.: Deletion predictions of MSplazer with support two and three, and the 454 analysis pipeline with variance setting of 1 bp position and 5% size variance (*pt1st5*): Shown are the total number of predictions, true positives (TP) and false positives (FP).

pt0st0	MSplazer support=2	MSplazer support=3	454 pipeline
Predictions	6,032	1,958	3,774
TP	4,660	1,621	2,685
FP	1,372	337	1,089

Table 5.9.: Deletion predictions of MSplazer with support two and three, and the 454 analysis pipeline with no allowed variance in position and size (*pt0st0*).

When we directly compare the stated PPVs (see Figure 5.4), we see that for the default variance setting (*pt1st5*, dark green), the 454 pipeline has the best PPV (93.4%). Not surprisingly, we found less predictions for MSplazer with support=3 than with support=2, but the PPV with support=3 is only a bit higher (87.3%, 84.2%). When we allow no variance (*pt0st0*, light green), MSplazer with support=3 now has the best PPV (82.8%), but also MSplazer with support=2 has a better PPV than the 454 pipeline (77.3%, 71.1%).

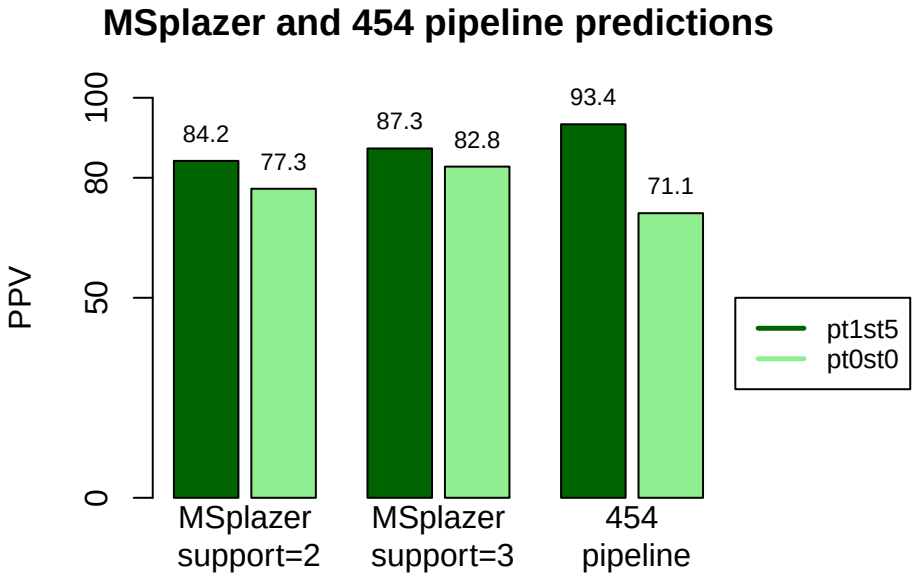


Figure 5.4.: MSplazer and 454 analysis pipeline PPVs for deletion predictions: PPV values are shown for variance setting of (1) 1 bp position variance and 5% deletion size variance (*pt1st5*, dark green), and (2) no allowed variance (*pt0st0*, light green).

Finally, we compare our results directly to the 454 pipeline (see Figure 5.5) with both the results from support two and three. We allow the same variance setting as for the comparison to the exon-exon junction reference set (*pt1st5*).

For MSplazer support=2, we find an overlap of 2,087 (35.6% MSplazer, 55.3% 454 pipeline). For MSplazer support=3, the overlap is 895 (23.7% 454 pipeline, 45.7% MSplazer). It is known that SV predictions methods often have a low overlap (see Alkan et al. (2011)). Since precision for both methods is high, the potential in finding new and good predictions is high for both methods. When we compare MSplazer with support=2 and the 454 pipeline allowing no variance, we still find a overlap of 1,488 deletions (24.7% MSplazer, 39.4% 454 pipeline), given the fact that the 454 pipeline had a much lower precision at *pt0st0* level on the exon-exon junction reference set.

All these values show, that our predictions are more precise than those from the 454 pipeline down to the exact breakpoint position. The precision increases with a

higher support, but then we miss many true positives. In the current version, we only consider the best chain for breakpoint predictions. A major improvement would be to take suboptimal chains, and even more, other reads at the same genomic location into consideration for determining the best chain. This should make the predictions much more precise even for a read support of two.

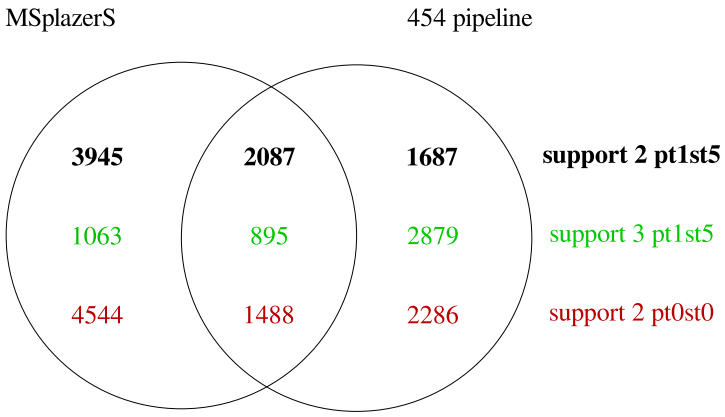


Figure 5.5.: Venn diagram of deletions predicted from MSplazer versus the 454 analysis pipeline: Position and size of the deletions are equal down to 1 bp position and 5% of the size (*pt1st5*, bold black and green) or are exactly the same (*pt0st0*, red). Read support for MSplazer is two (bold black and red) or three (green).

5.6. Discussion

We tested our method using data from chimeric transcript sequences, contigs from simulated Illumina reads, and real 454 human RNA-Seq reads. A parameter and runtime analysis showed that Stellar is indeed the major runtime factor, and that the overlap parameter is the most influencing parameter which supports our approach. However, within the scope of this thesis, it was sufficient to manually split up the reads or contigs to run them in parallel.

We successfully applied our method to chimeric transcript sequences, finding the correct breakpoints of inversions, deletions and inter-chromosomal translocations even for multiple splits.

For the contig set, we compared our predictions with a reference set containing the simulated SVs. MSplazer without the small Stellar indels yielded high recall and very high positive predicted values for all insertions <300 bp, inversions, duplications, and for deletions <10,000 bp in size. For deletions larger than that, MSplazer found all three deletions in the reference set, but also predicted many more false positives (PPV 0.03 %) which may indicate a need to penalise very large deletions.

Insertions that are not included in the contig are not reported in the current version of MSplazer. They most likely result in partial chains. Evaluating partial chains, we

could predict at least the boundaries of possible large insertions. Determining precise breakpoints and content however results in an insertion assembly problem.

Compared to Grouper and SVDetect, MSplazer yields similar or even better PPVs for deletions and insertions but has a lower recall than Grouper. SVDetect cannot detect insertions, and did not predict any deletions <300 bp. For duplications, both PPV and recall are similarly high. Grouper cannot detect duplications >10,000 bp. As for inversions, MSplazer has a much better recall than both tools and also a similar PPV as SVDetect. These values show that our method can compete with both tools but also show room for improvement.

To generally increase the number of predictions, we could iteratively call Stellar with a smaller minimum length for reads or contigs, where MSplazer did not find any match in the first place. However, with increasing read length, we expect our method to find more predictions. Illumina, e.g., just announced 250 bp length paired-end reads (Illumina Inc., 2012).

When taking into account the small indels, we could increase the recall value but lost precision at the same time. In the current version, these indels have the same support parameter as the larger SVs, which was set to one for this dataset. It would probably make more sense to separate both supports and then generally increase the support for the smaller indels. The predictions of the small indels then need to be evaluated further, so there is still room for improvement here.

For the 454 RNA-Seq reads, we validated our deletion predictions against a reference set containing all exon-exon junctions derived from RefSeq. Precision was indicated by the percentage of predictions confirmed by the reference set, and the predictions were compared to the results from the 454 analysis pipeline. MSplazer, given a read support of two, found many more deletions than the 454 pipeline with an even higher precision when we allow no variance in position or size of the deletion.

To further improve predictions, we could take additional information into account. For example, we could favour edges in the graph that confirm insert sizes from paired-end data or a preferred position range to overcome ambiguity of matches and chains, or favour edges that match a predefined SV type.

In the current version, we only consider the best chain of each read for breakpoint predictions. A major improvement would be to take suboptimal chains, and even more, other reads at the same genomic location into consideration for determining the best chain. This should make the predictions much more precise, and would also benefit the small indel call from the Stellar matches.

6. Conclusion and Future Work

With the advance of NGS, reads become more likely to cross breakpoints resulting from SVs or exon-exon junctions. Existing SV detection methods typically focus on the detection of a particular type of SV. One of the most promising generic approaches is the split-read approach since it allows breakpoint detection at base-pair resolution. Existing split-read methods are based on a global alignment approach of artificial splits of a read, and often have additional constraints on collinearity and location of the splits.

We designed and implemented a generic multi-split chaining method that is able to detect all types of SVs including inversions, inter- and intra-chromosomal translocations, duplications, insertions, and deletions. Our method uses a split-read approach based on local matches of the read which allows for arbitrary number and location of the splits. In addition, we do not set constraints on match orientation or location, and thereby allow for the SV types named above.

We successfully tested our approach on both simulated Illumina contig data and real 454 RNA-Seq data, demonstrating its platform independence. For the Illumina contigs, we validated our results with a reference set containing the simulated SVs, and yielded high recall and precision values. For the 454 RNA-Seq reads, we validated our deletion predictions against a reference set containing all exon-exon junctions derived from RefSeq. Given a read support of two, MSplazer found many more exon-exon junctions than the 454 analysis pipeline with an even higher precision when we require exact detection. Our results can compete with the results of the tool SVDetect and the Illumina analysis software on the same contig data set, and with the 454 analysis software on the RNA-Seq dataset. However, there is still room for improvement.

Future Work

From the memory and computation time point of view, Stellar is the bottle neck of our approach. For the Illumina and 454 test datasets, Stellar needed several hours for a batch of 5000 contigs or reads. For larger datasets or extended use of our method, this makes usage quite laborious. To improve runtime, one approach could be to enhance Stellar to process all the reads parallel. Another way would be to wrap the usage of Stellar where read files are automatically split. However, automatic splitting files could be more prone to errors.

We still need to investigate our penalty parameters for different SV types on the edges in the graph. For example, the very low PPV for large deletions (PPV 0.03 %) may indicate a need to penalise deletions when they are huge. Generally, the penalties are

probably dependent on the organism and frequency of insertions, deletions, translocations and inversions.

In the current implementation, partial chains are not evaluated. Partial chains could contain information about possible boundaries of larger insertions which could be useful for a following insertion assembly step.

A general extension for compatibility determination would be to take additional information into account. For paired-end data, we would concatenate the pairs into one read what produces edges belonging to the concatenation position. Edges that confirm the insert size of the pair would be favoured over edges which do not. Other additional information could be assumptions of SV types for individual reads (e.g. splice sites in the case of RNA-Seq data), or a preferred position range to overcome ambiguity.

During match compatibility computation, we also allow a small gap between matches which accounts for the fact that larger SVs are often accompanied by a micro-indel. At the moment, this indel is not refined and reported since it is covered by the larger SV. This refinement would be a good functional extension to the method.

As seen in Figure A.7, there are cases where we have multiple best chains per read. In the current version, our method arbitrarily chooses one of them. A major improvement to the method would be to take suboptimal chains and also chains from other reads in the same genomic location into account when determining the best chain of a read. This may give the true breakpoint a higher support and at the same time reduce the number of false positive predictions.

Acknowledgements

First of all, I want to thank my supervisor, Prof. Knut Reinert, for giving me the opportunity and time to work on this project. I thank Tobias Rausch for his interest and willingness to review my reseach work.

Special thanks to my supervisor Anne-Katrin Emde, who guided and supported me through the entire process.

I am grateful to the partners from Illumina for giving the permission to use the Illumina contig dataset and their results from Grouper and SVDetect, and to Magdalena Feldhahn for providing us with the 454 RNA-Seq dataset and the results from the 454 analysis pipeline.

Many thanks to my colleagues from the Algorithmic Bioinformatics Group and to Edzard Höfig, who supported me with comments, ideas and enlightening coffee breaks.

Special thanks also to René Rahn, Bastian Kayser and Uwe Kamper for their extensive proof reading and support.

A. Appendix

A.1. Proof of Concept and Qualitative Analysis Using Sequences of Chimeric Transcripts

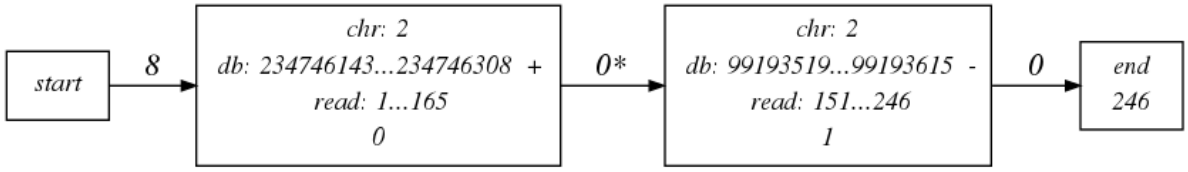


Figure A.1.: INPP4A-HJURP FJ423742: Compatibility graph for Stellar matches with $min.length = 30$.

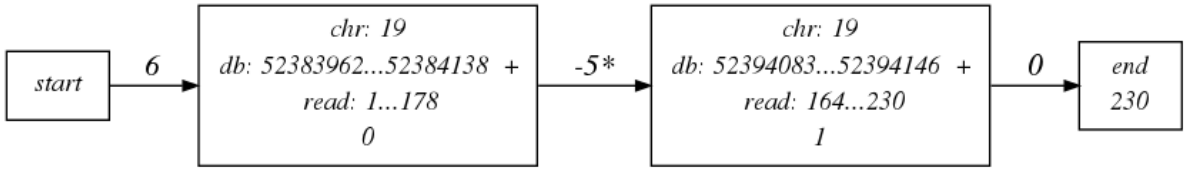


Figure A.2.: ZNF649-ZNF577 FJ423743: Compatibility graph for Stellar matches with $min.length = 30$.

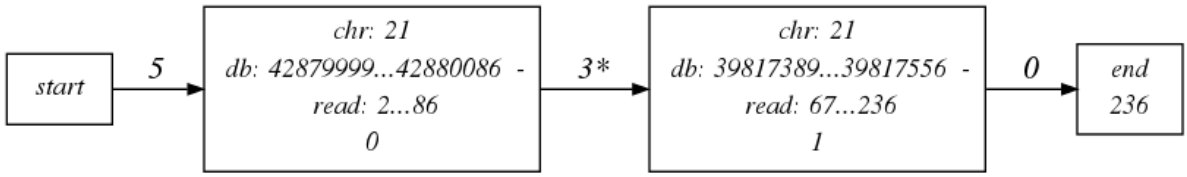


Figure A.3.: TMPRSS2-ERG FJ423744: Compatibility graph for Stellar matches with $min.length = 30$.

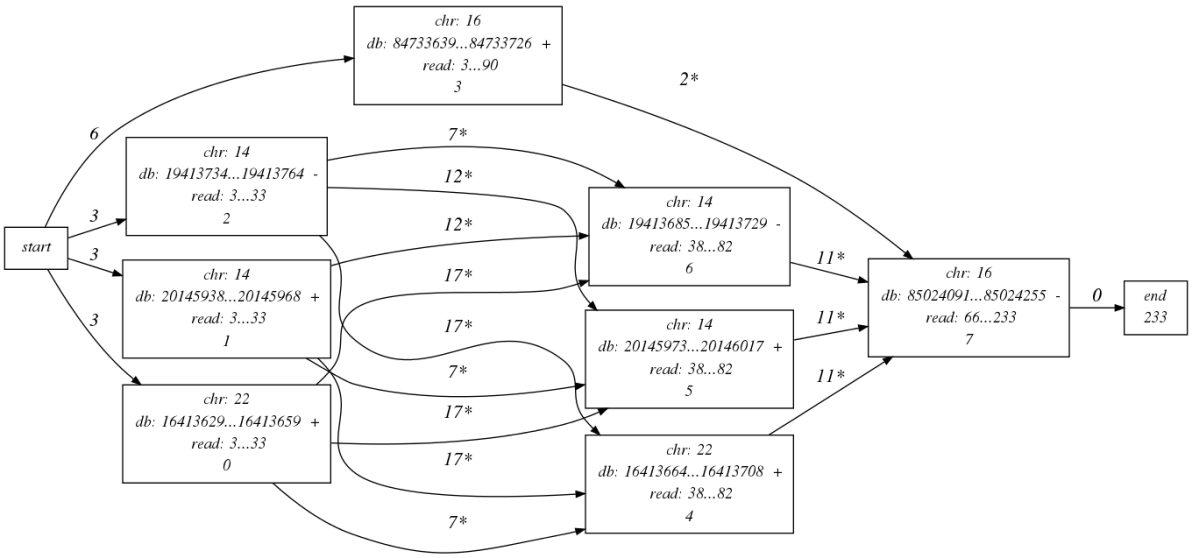


Figure A.4.: USP10-ZDHHC7 FJ423745: Compatibility graph for Stellar matches with $min.length = 30$.

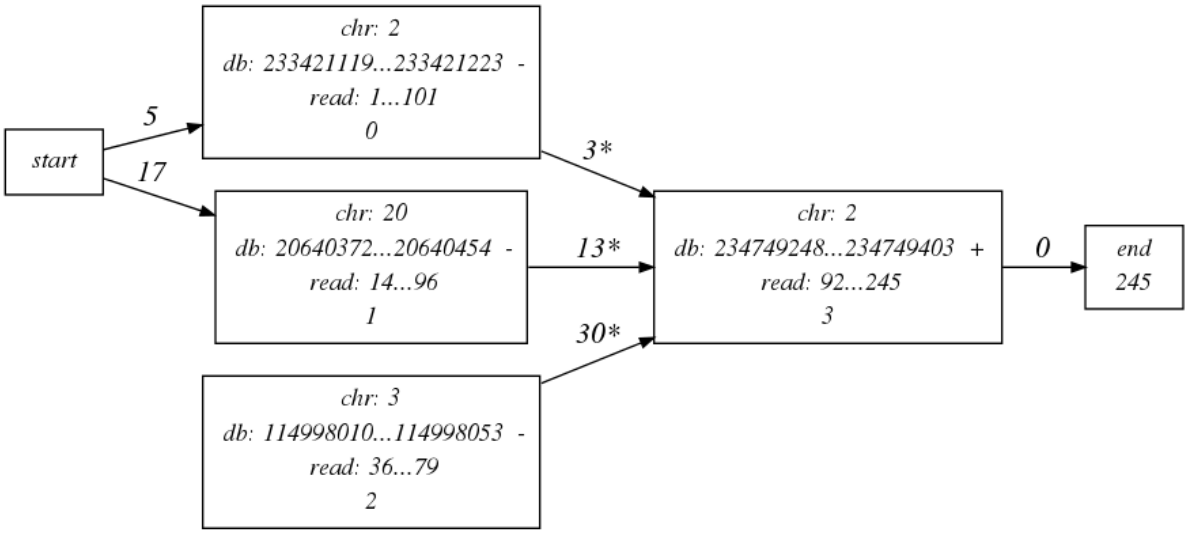


Figure A.5.: HJURP-EIF4E2 FJ423746: Compatibility graph for Stellar matches with $min.length = 30$.

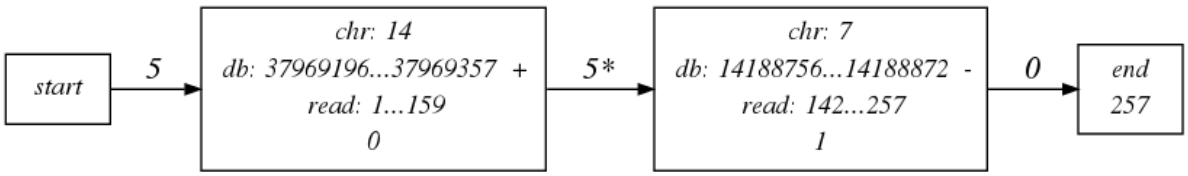


Figure A.6.: MIPOL1-DGKB FJ423747: Compatibility graph for Stellar matches with $min.length = 30$.

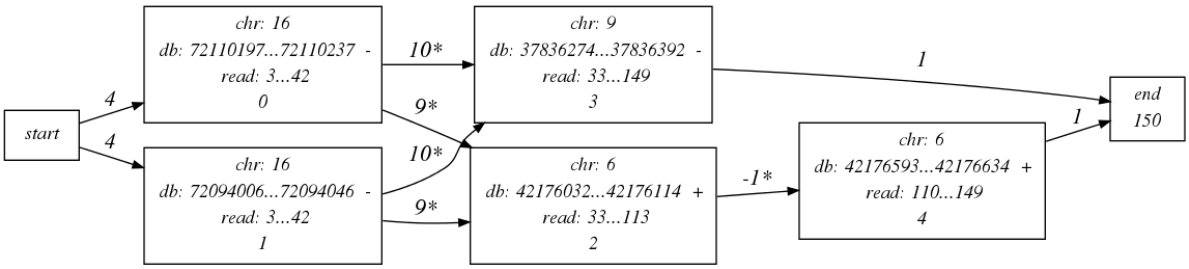


Figure A.7.: MRPS10-HPR FJ423748: Compatibility graph for Stellar matches with $min.length = 30$.

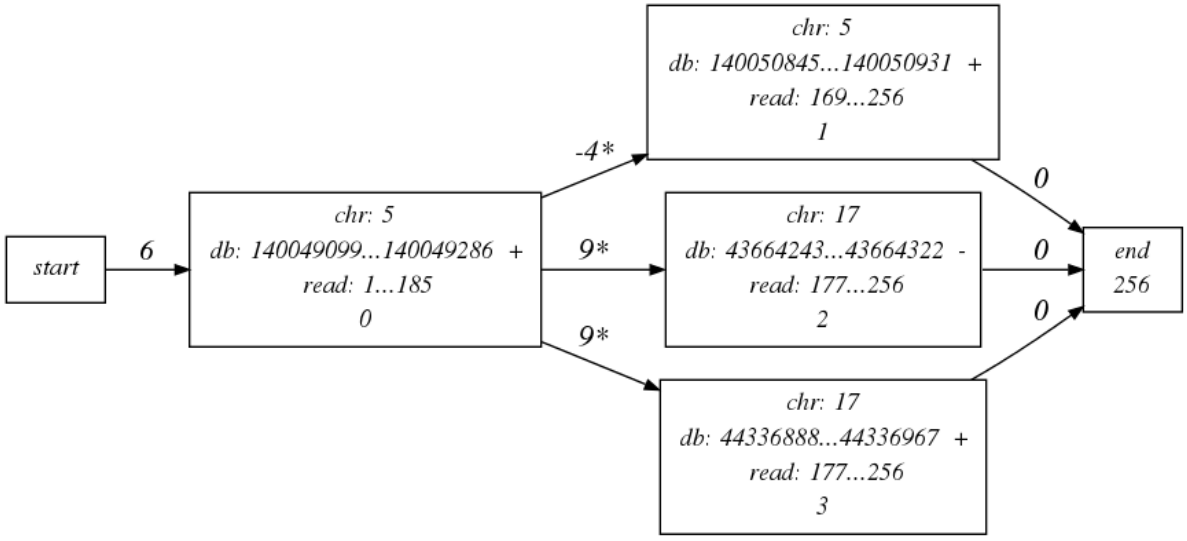


Figure A.8.: WDR55-DND1 FJ423749: Compatibility graph for Stellar matches with $min.length = 30$.

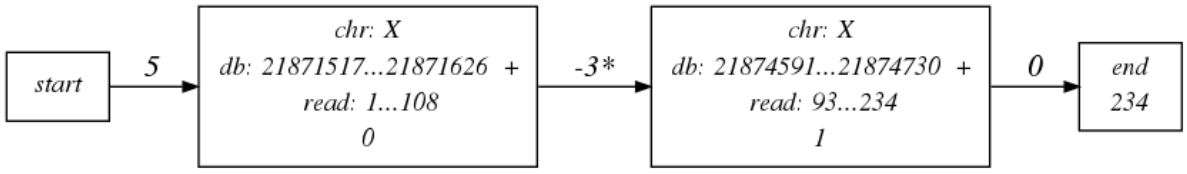


Figure A.9.: MBTPS2-YY2 FJ423751: Compatibility graph for Stellar matches with $min.length = 30$.

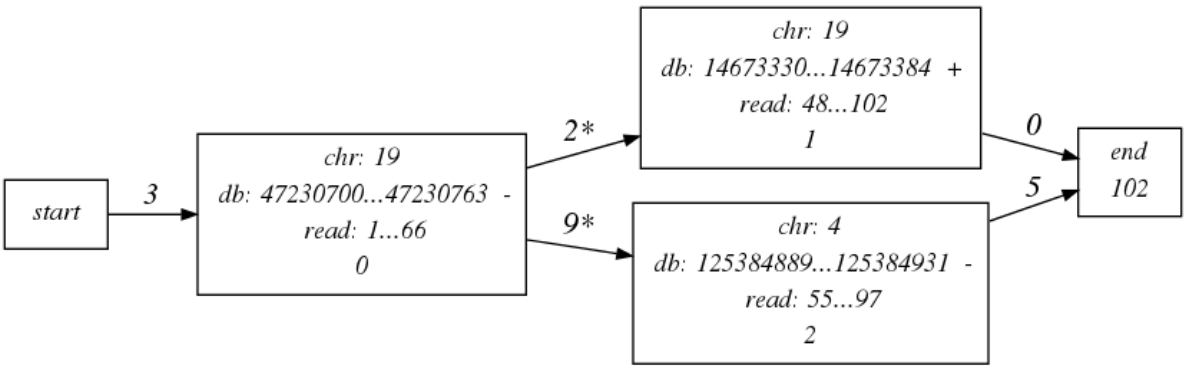


Figure A.10.: STRN4-GPSN2 FJ423752: Compatibility graph for Stellar matches with $min.length = 30$.

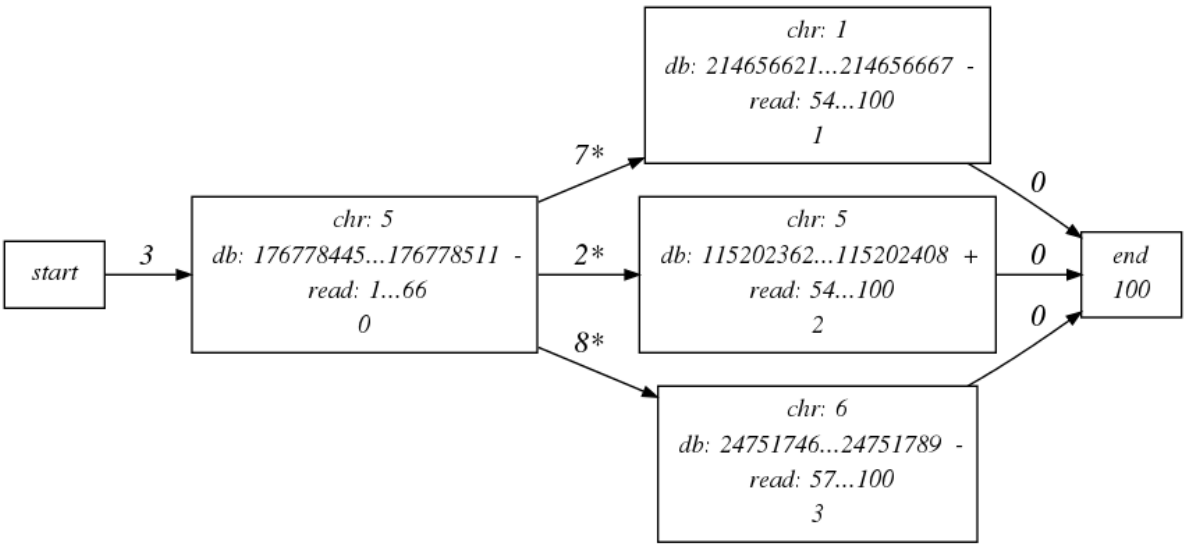


Figure A.11.: LMAN2-AP3S1 FJ423753: Compatibility graph for Stellar matches with $min.length = 30$.

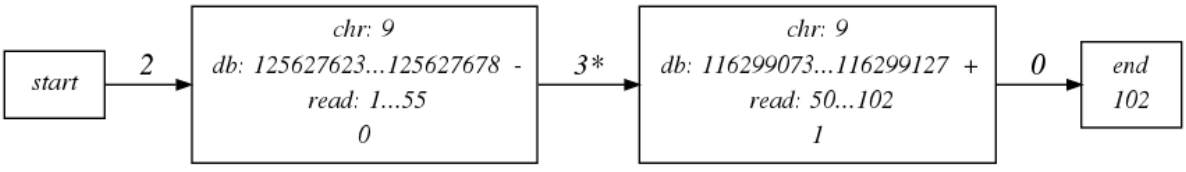


Figure A.12.: RC3H2-RGS3 FJ423754: Compatibility graph for Stellar matches with $min.length = 30$.

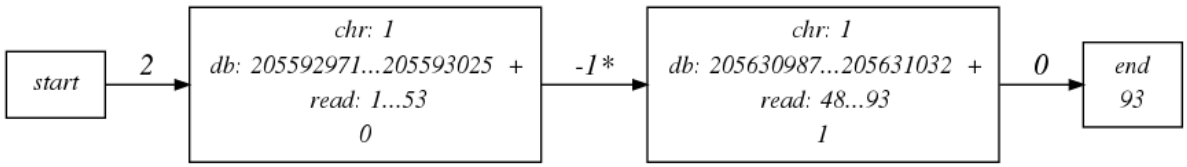


Figure A.13.: SLC45A3-ELK4 FJ423755: Compatibility graph for Stellar matches with $min.length = 30$.

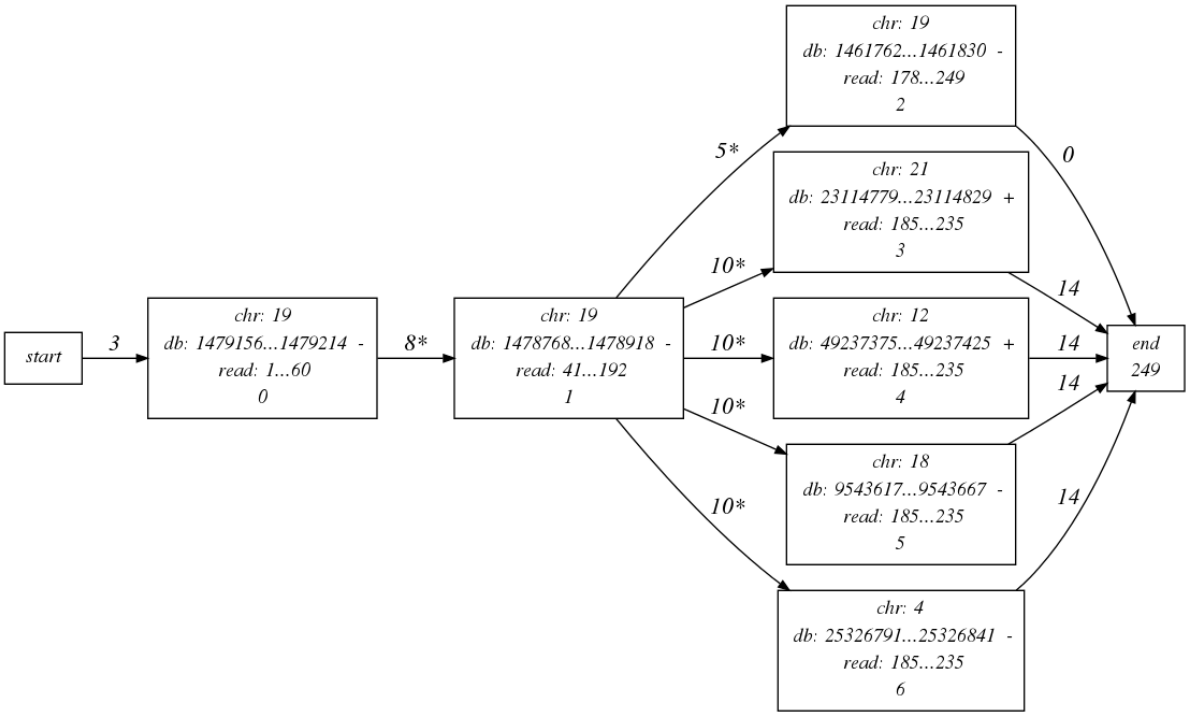


Figure A.14.: C19orf25-APC2 (Intron) FJ423750: Compatibility graph for Stellar matches with $min.length = 50$.

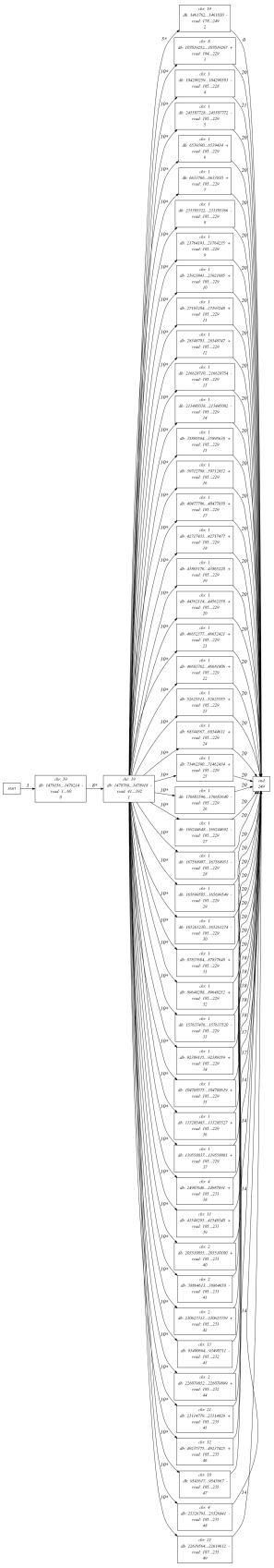


Figure A.15.: C19orf25-APC2 (Intron) FJ423750: Compatibility graph for Stellar matches with $min.length = 30$.

Bibliography

Abyzov, A. and Gerstein, M. (2011), 'AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision', *Bioinformatics* **27**(5), 595–603.

Alkan, C., Coe, B. P. and Eichler, E. E. (2011), 'Genome structural variation discovery and genotyping', *Nature Reviews Genetics* **12**, 363–376.

Au, K. F., Jiang, H., Lin, L., Xing, Y. and Wong, W. H. (2010), 'Detection of splice junctions from paired-end RNA-seq data by SpliceMap', *Nucleic Acids Research* **38**, 4570–4578. `www.stanford.edu/group/wonglab/SpliceMap`.

Chen, K., Wallis, J. W., McLellan, M. D. et al. (2009), 'BreakDancer: an algorithm for high-resolution mapping of genomic structural variation', *Nature Methods* **6**(9), 677–684.

Cormen, T. (2001), *Introduction to algorithms*, MIT electrical engineering and computer science series, MIT Press.

Döring, A., Weese, D., Rausch, T. and Reinert, K. (2008), 'SeqAn an efficient, generic C++ library for sequence analysis.', *BMC Bioinformatics* **9**, 11.

Emde, A.-K., Schulz, M. H., Weese, D., Sun, R., Vingron, M., Kalscheuer, V. M., Haas, S. A. and Reinert, K. (2012), 'Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS', *Bioinformatics* pp. 1–9. `http://www.seqan.de/projects/splazers.html`.

Feldhahn, M., Menzel, M., Weide, B., Bauer, P., Meckbach, D., Garbe, C., Kohlbacher, O. and Bauer, J. (2011), 'No evidence of viral genomes in whole-transcriptome sequencing of three melanoma metastases.', *Exp Dermatol* **20**(9), 766–768.

Feuk, L., Carson, A. R. and Scherer, S. (2006), 'Structural variation in the human genome.', *Nature Reviews Genetics* **7**, 85–97.

Franklin, R. E. and Gosling, R. G. (1953), 'Molecular configuration in sodium thymonucleate.', *Nature* **171**(4356), 740–741.

Gogol-Döring, A. and Reinert, K. (2009), *Biological sequence analysis using the SeqAn C++ library*, Chapman and Hall/CRC mathematical & computational biology series, CRC Press.

Illumina Inc. (2012). http://investor.illumina.com/phoenix.zhtml?c=121127&p=irol-newsArticle&ID=1646757&highlight=.

Iyer, M. K., Chinnaiyan, A. M. and Maher, C. A. (2011), 'ChimeraScan: a tool for identifying chimeric transcription in sequencing data.', *Bioinformatics* **27**(20), 2903–2904.

Karakoc, E., Alkan, C., O'Roak, B. J., Dennis, M. Y., Vives, L., Mark, K., Rieder, M. J., Nickerson, D. A. and Eichler, E. E. (2011), 'Detection of structural variants and indels within exome data.', *Nat Methods* **9**(2), 176–178.

Kehr, B., Weese, D. and Reinert, K. (2011), STELLAR: fast and exact local alignments, *in* 'Ninth Annual Research in Computational Molecular Biology (RECOMB) Satellite Workshop on Comparative Genomics', Vol. 12(Suppl 9). http://www.seqan.de/projects/stellar.html.

Krawitz, P., Rödelsperger, C., Jäger, M., Jostins, L., Bauer, S. and Robinson, P. N. (2010), 'Microindel detection in short-read sequence data.', *Bioinformatics* **26**(6), 722–729.

Lam, H. Y. K., Mu, X. J., Stütz, A. M., Tanzer, A., Cayting, P. D., Snyder, M., Kim, P. M., Korbel, J. O. and Gerstein, M. B. (2010), 'Nucleotide-resolution analysis of structural variants using BreakSeq and a breakpoint library', *Nature Biotechnology* **28**(1), 47–57.

Lander, E. S., Linton, L. M., Birren, B. et al. (2001), 'Initial sequencing and analysis of the human genome.', *Nature* **409**(6822), 860–921.

Langmead, B., Trapnell, C., Pop, M. and Salzberg, S. L. (2009), 'Ultrafast and memory-efficient alignment of short DNA sequences to the human genome', *Genome Biology* **10**(3), R25.

Li, H. and Durbin, R. (2010), 'Fast and accurate long-read alignment with Burrows-Wheeler transform', *Bioinformatics* **26**(5), 589–595.

Li, H. and Homer, N. (2010), 'A survey of sequence alignment algorithms for next-generation sequencing.', *Brief Bioinform* **11**(5), 473–483.

Maher, C. A., Kumar-Sinha, C., Cao, X., Kalyana-sundaram, S., Han, B., Jing, X., Sam, L., Barrette, T., Palanisamy, N. and Chinnaiyan, A. M. (2009), 'Transcriptome sequencing to detect gene fusions in cancer', *Nature Letters* **458**.

Medvedev, P., Stanciu, M. and Brudno, M. (2009), 'Computational methods for discovering structural variation with next-generation sequencing', *Nature Methods* **6**(11), S13–S20.

Metzker, M. L. (2010), 'Sequencing technologies - the next generation.', *Nat Rev Genet* **11**(1), 31–46.

Mills, R. E., Luttig, C. T., Larkins, C. E., Beauchamp, A., Tsui, C., Pittard, W. S. and Devine, S. E. (2006), 'An initial map of insertion and deletion (INDEL) variation in the human genome', *Genome Research* **16**, 1182–1190.

Mills, R. E., Walter, K., Stewart, C. et al. (2011), 'Mapping copy number variation by population-scale genome sequencing', *Nature* **470**, 59–65.

Pruitt, K. D., Tatusova, T., Klimke, W. and Maglott, D. R. (2009), 'NCBI Reference Sequences: current status, policy and new initiatives.', *Nucleic Acids Res* **37**(Database issue), D32–D36.

Roche Diagnostics Corporation (2012). `http://my454.com/products/analysis-software/index.asp`.

Schuster, S. C. (2008), 'Next-generation sequencing transforms today's biology.', *Nat Methods* **5**(1), 16–18.

Sebat, J., Lakshmi, B., Malhotra, D. et al. (2007), 'Strong Association of De Novo Copy Number Mutations with Autism', *Science* **316**(5823), 445–449.

Stankiewicz, P. and Lupski, J. R. (2010), 'Structural variation in the human genome and its role in disease.', *Annu Rev Med* **61**, 437–455.

The 1000 Genomes Project Consortium (2010), 'A map of human genome variation from population-scale sequencing.', *Nature* **467**, 1061–1073.

Tuzun, E., Sharp, A. J., Bailey, J. A. et al. (2005), 'Fine-Scale structural variation of the human genome', *Nat.Genet.* **37**(7), 727–732.

Venter, J. C., Adams, M. D., Myers, E. W. et al. (2001), 'The sequence of the human genome.', *Science* **291**(5507), 1304–1351.

Watson, J. D. and Crick, F. H. (1953), 'Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid.', *Nature* **171**(4356), 737–738.

Weese, D., Emde, A.-K., Rausch, T., Döring, A. and Reinert, K. (2009), 'RazerS-Fast read mapping with sensitivity control', *Genome Research* **19**, 1646–1654.

Xi, R., Kim, T.-M. and Park, P. J. (2011), 'Detecting structural variations in the human genome using next generation sequencing', *Briefings in Functional Genomics* **9**(5), 405–415.

Ye, K., Schulz, M., Long, Q., Apweiler, R. and Ning, Z. (2009), 'Pindel: pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads.', *Bioinformatics* **25**, 2865–2871.

Zeitouni, B., Boeva, V., Janoueix-Lerosey, I., Loeillet, S., Legoix-nÃ©, P., Nicolas, A., Delattre, O. and Barillot, E. (2010), 'SVDetect: a tool to identify genomic structural variations from paired-end and mate-pair sequencing data.', *Bioinformatics* **26**(15), 1895–1896.

Zhang, J. and Wu, Y. (2011), 'SVseq: an approach for detecting exact breakpoints of deletions with low-coverage sequence data', *Bioinformatics* .

Zhang, Z. D., Du, J., Lam, H., Abyzov, A., Urban, A. E., Snyder, M. and Gerstein, M. (2011), 'Identification of genomic indels and structural variations using split reads', *BMC Genomics* **12**(375).