ALGORITHMS TO RESOLVE LARGE SCALE AND COMPLEX STRUCTURAL VARIANTS IN THE HUMAN GENOME

by

MATTHEW HAYES

Submitted in partial fulfillment of the requirements
For the Degree of Doctor of Philosophy

Dissertation Advisor: Jing Li

Department of Electrical Engineering and Computer Science
CASE WESTERN RESERVE UNIVERSITY

August, 2013
We hereby approve the thesis/dissertation of

Matthew Elliot Hayes

______________________________________________________

Doctor of Philosophy
candidate for the ________________________________degree *.

Jing Li

______________________________________________________

Mehmet Koyuturk

______________________________________________________

Guo-Qiang Zhang

______________________________________________________

Angela Ting

______________________________________________________

6/19/2013
(date) __________________________

*We also certify that written approval has been obtained for any proprietary material contained therein.
To my grandparents Willis and Ruby Archon, and my great grandparents Clarence and Clara Moore.
# Table of Contents

List of Tables vi

List of Figures viii

Acknowledgments xi

Abstract xii

Chapter 1. Introduction 1

  1.1 Motivation .............................................................. 1
  1.2 Objective ............................................................. 3
  1.3 Contribution .......................................................... 4
  1.4 Organization of thesis .............................................. 5

Chapter 2. Background 6

  2.1 The array CGH platform ............................................. 6
  2.2 Next-generation sequencing platforms ............................. 8
    2.2.1 General steps in NGS sequencing experiments ................. 9
  2.3 Structural variation ................................................ 11
    2.3.1 Basic types ...................................................... 12
      2.3.1.1 Deletions ................................................. 12
      2.3.1.2 Insertions ............................................... 13
      2.3.1.3 Inversions ............................................... 15
      2.3.1.4 Tandem repeats ......................................... 16
      2.3.1.5 Translocations .......................................... 17
    2.3.2 Complex structural variants ................................ 18
      2.3.2.1 Balanced closed chain rearrangements .................. 19
      2.3.2.2 Chromothripsis and double minute chromosomes ....... 19
  2.4 Detecting structural variants using NGS ......................... 22
# Chapter 4. Algorithms

4.1 Detecting copy number change points in aCGH data using log ratio triangulation

4.1.1 Motivation

4.1.2 Methods

4.1.2.1 Log-2 ratio triangulation

4.1.2.2 The algorithm

4.1.2.3 Algorithm to calculate \( \gamma \)

4.1.2.4 Proof of time complexity

4.1.3 Experiments

4.1.3.1 Study on the Coriell Cell Line

4.1.3.2 Results: Coriell Cell Line

4.1.3.3 Study on simulated dataset

4.1.3.4 Results: simulated data

4.1.4 Discussion

4.2 Detecting large structural variants at base pair level in NGS data

4.2.1 Methods

4.2.1.1 Requirements of a valid discordant read pair cluster

4.2.1.2 Interchromosomal variants

4.2.1.3 Chimeric breakpoint classification

4.2.1.4 Deletions

4.2.1.5 Inversions

4.2.1.6 Tandem repeats

4.2.1.7 Double minute chromosomes

4.2.2 Experimental design

4.2.2.1 Interchromosomal variants: two simulated datasets

4.2.2.2 Interchromosomal variants: primary prostate cancer datasets

4.2.2.3 Deletions

4.2.2.4 Inversions

4.2.2.5 Tandem repeats

4.2.2.6 Double minute chromosomes
4.2.3 Results

4.2.3.1 Interchromosomal variants: results on simulated datasets

4.2.3.2 Interchromosomal variants: results on prostate cancer datasets

4.2.3.3 Deletions: results on chromosome 9 simulated data

4.2.3.4 Deletions: results on prostate cancer datasets

4.2.3.5 Inversions: results on prostate cancer datasets

4.2.3.6 Tandem repeat breakpoints: results on prostate cancer datasets

4.2.3.7 Double minute chromosomes: results on simulated data

4.2.4 Discussion

Chapter 5. Conclusion and Future Work

Appendix

Bibliography
List of Tables

4.1 For $c = 2$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds. ........................................... 38
4.2 For $c = 3$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds. ........................................... 38
4.3 Results of the CBS algorithm on Coriell dataset. Results are consistent with those in the study of Olshen et al. ................................. 38
4.4 For $c = 2$. Results of Study on Simulated Data. Each test run required approximately 5m30s. ......................................................... 41
4.5 For $c = 3$. Results of Study on Simulated Data. Each test run required approximately 5m30s. ......................................................... 42
4.6 Structural variants inserted into the first simulated dataset. U = unbalanced translocation, II = interchromosomal insertion, and B = balanced translocation. For the “II” and “B” variants, the partner breakpoints are listed consecutively. For “II” variants, the donor chromosome and its breakpoint are bolded. Note that the chr3 and chr6 balanced translocation contains a 1000-bp duplication, so it is not entirely reciprocal. ......................................................... 60
4.7 Structural variants inserted into the second simulated dataset. ................................. 61
4.8 Summary of deletions inserted into chromosome 9. There were a total of 102 simulated deletions. ......................................................... 63
4.9 Simulated dataset 1 results (100 bp reads) ......................................................... 67
4.10 Simulated dataset 2 results (75 bp reads) ......................................................... 67
4.11 Results on the PR-0508 dataset. ......................................................... 71
4.12 Results on the PR-1783 dataset. ......................................................... 71
4.13 Sensitivity and specificity on the simulated dataset. BD = BreakDancer, PEG = Pegasus, CRST = CREST. Some methods have redundant predictions, but these were not counted against their specificity. 72
4.14 F-scores and average breakpoint error for each method on the simulated datasets. ......................................................... 73
4.15 Sensitivity and specificity results on PR-0508. ......................................................... 74
4.16 F-score and breakpoint error on PR-0508. ......................................................... 74
4.17 Sensitivity and specificity results on PR-1783. ......................................................... 74
4.18 F-score and breakpoint error on PR-0508. . . . . . . . . . . . . . . . 74
4.19 PR-0508 results. Inversion breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm. . . . . . . . . . . . . . . 75
4.20 PR-1783 results. Inversion breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm. . . . . . . . . . . . . . . 76
4.21 PR-0508 results. Tandem repeat breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm. . . . . . . . . . . . . . . 77
4.22 PR-1783 results. Tandem repeat breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm. . . . . . . . . . . . . . . 77
List of Figures

2.1 Sample output from an aCGH experiment. .......................................................... 7
2.2 NGS sequencing step 1: isolate DNA. ................................................................. 10
2.3 NGS sequencing step 2: create fragments through random shearing. ................. 10
2.4 NGS sequencing step 3a: sequencing of both ends of sheared fragments (for paired-end sequencing). ................................................................. 11
2.5 NGS sequencing step 3b: sequencing of one end of a sheared fragment (for single-end sequencing). ................................................................. 11
2.6 A deletion from a chromosome i. ............................................................................. 13
2.7 A gene fusion caused by a deletion. The genomic segment in the top image has a deletion represented by the green box. As a result, genes A and B are now fused together. ................................................................. 13
2.8 Structural variation: insertions. ................................................................................ 14
2.9 A gene fusion caused by an interchromosomal insertion. Gene A and gene B are located on separate, non-homologous chromosomes. A segment from gene B has been inserted into a region of gene A, thus creating a fusion. ................................................................. 14
2.10 Structural variation: inversions. The colored circles represent DNA sequences. When an inversion occurs, this segment will change its orientation. ................................................................. 15
2.11 Gene fusion caused by an inversion. The segment within the green box is inverted, causing the fusions depicted in the bottom image. ................................. 16
2.12 Structural variation: tandem repeat. ................................................................. 17
2.13 Structural variation: non-reciprocal translocation. ............................................. 18
2.14 Structural variation: reciprocal translocation. ..................................................... 18
2.15 Complex SV: balanced closed chain rearrangement. There is a perfect balanced exchange of arms amongst the three chromosomes. ......................... 20
2.16 Complex SV: chromothripsis. During cancer development, the genome may shatter in a single catastrophic event. ............................................ 21
2.17 Erroneous repair of a genomic region affected by chromothripsis. The segments comprising the genome subset may be highly rearranged after the event. It is even possible to see the formation of double minutes. 21
2.18 Complex SV: double minute chromosomes. Double minutes tend to contain oncogenes. They are also highly amplified.

2.19 Structural variants and the discordant alignments that support them. The colored circles for the inversion case represent DNA sequences in the affected segment. Most NGS-based algorithms predict structural variants by searching for these anomalously mapped read pairs. The “D” represents the donor genome, and the “R” represents the reference genome.

4.1 Illustrative example of $T, F,$ and $H$ for a given triangle. The points $x_1$, $x_2$, and $x_3$ are log-2 ratio values in the aCGH data, and triangles are created for every three consecutive log-2 ratio values.

4.2 CGH-Triangulator algorithm.

4.3 Translocation captured by a cluster of three chimeric read pairs. The first set of reads map to the forward strand of chromosome $i$, and the second set map to reverse strand of chromosome $j$. The distance between the outermost reads to the breakpoint are $D_1$ and $D_2$, for chromosomes $i$ and $j$ respectively. These distances must be less than or equal to $\text{mean} + k \times \text{stdev}$.

4.4 The three types of translocations and the mapping orientations that result when a pair spans the breakpoint. These orientations assume that Illumina technologies were used in sequencing. Bellerophon deduces the type of fusion based on the mapping orientation of the pairs in a cluster.

4.5 The formation of soft-clipped reads. Soft-clipped reads span the translocation boundary between chromosomes $i$ and $j$. As a result, these reads may align partially to chromosome $i$ and partially to chromosome $j$.

4.6 The uncertainty of chimeric breakpoint prediction. Many SV methods only attempt to find chimeric boundaries as shown above, but they do not attempt to classify them. The above event could denote a balanced translocation between chromosomes 5 and 9, or it could denote an interchromosomal insertion from chromosome 9 to 5.

4.7 An interchromosomal insertion and its resulting mapping signatures. Chromosome $i$ donated a segment to chromosome $j$. The read pair in chromosome $i$ spans the donor site, and when it is mapped back to the reference, its mapped distance will be much greater than its original distance in sequencing. To infer an interchromosomal insertion, Bellerophon not only searches for mirroring records, but it searches for these long, anomalously mapped pairs (AP). If two mirroring records have no corresponding long APs, then the records could imply a balanced translocation.
4.8 An interchromosomal insertion and its resulting mapping signatures. Chromosome $i$ donated a segment to chromosome $j$. The read pair in chromosome $i$ spans the donor site, and when it is mapped back to the reference, its mapped distance will be much greater than its original distance in sequencing. To infer an interchromosomal insertion, Bellerophon not only searches for mirroring records, but it searches for these long, anomalously mapped pairs (AP). If two mirroring records have no corresponding long APs, then the records could imply a balanced translocation. 

4.9 Algorithm to build the amplicon graph $G$ and auxiliary graph $H$. 

4.10 Algorithm to find all double minutes that are represented in the amplicon graph. 

4.11 Conversion of NGS breakpoints and amplicons to an amplicon graph. The segments in the top image are amplicons with elevated copy number. The colored lines represent SV predictions for a translocation (green line), and a deletion (blue line). The corresponding amplicon graph is depicted in the bottom image. Vertices are amplicons and edges are NGS-based SV breakpoint predictions. 

4.12 Visual results for the 100 bp simulated dataset. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions. 

4.13 Visual results for the 75 bp simulated dataset. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions. 

4.14 Visual results for PR-0508. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions. 

4.15 Visual results for PR-1783. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions. 

1 Flowchart for the interchromosomal insertion prediction algorithm. For mirroring records, Bellerophon first tries to classify them as interchromosomal insertions. If it cannot, it proceeds to the balanced translocation classification step depicted in Fig. 1. 

2 Flowchart for the balanced translocation prediction algorithm. Bellerophon allows for deletions and insertions of at most 1 Mb at the breakpoints. Bellerophon will pair two mirroring records whose breakpoints are closest. Although not depicted in the figure, Bellerophon allows for a maximum of two balanced translocation predictions between a pair of chromosomes. This allows our method to predict reciprocal translocations between two chromosomes that occur on both haplotypes.
Acknowledgments

I would like to thank my advisor Dr. Jing Li for his mentorship and guidance through the past five years. I would also like to thank Dr. Gultekin Ozsoyoglu for sponsoring my GAANN fellowship and for being a constant source of wisdom. I also thank my former lab mate Yoon Soo Pyon for his collaborative help on our earlier projects. Furthermore, I would like to thank Dr. Angela Ting, Dr. Mehmet Koyuturk, and Dr. Guo-Qiang Zhang for serving on my thesis committee. I would also like to thank my girlfriend Danielle St. Julien for her love and support through this process. Furthermore, I would like to thank my sister Claire Hayes for her support as well. Finally, I would like to thank my parents, Phebe and Harold Hayes for their constant love and unwavering support throughout my years of education.
Algorithms to Resolve Large Scale and Complex Structural Variants in the Human Genome

Abstract

by

MATTHEW HAYES

It has been shown that large scale genomic structural variants (SV) are closely associated with disease onset. In particular, the presence of these abnormalities may contribute to the onset and susceptibility of cancer through various mechanisms. Knowing the location and type of these variants can assist medical researchers in making insights into methods for diagnosis and treatment. It is also important to develop efficient methods to locate these variants. This thesis presents several algorithms for identifying and characterizing structural variants using array comparative genomic hybridization (aCGH) and high throughput next-generation sequencing (NGS) platforms. The aCGH-based algorithm (CGH-Triangulator) is considerably faster than a state-of-the-art method for identifying change points in aCGH data, and it has greater prediction power on datasets with low-to-moderate levels of noise. The NGS-based algorithms include methods to identify basic SV types, including deletions, inversions, translocations, and tandem repeats. They also include methods to identify double minute chromosomes, which are more complex structural variants. These methods use a hybrid strategy to identify variants at base-pair resolution. Using two primary
prostate cancer datasets and simulated datasets, we compared our methods to previously published NGS algorithms. Overall, our methods had favorable performance with respect to breakpoint prediction accuracy, sensitivity, and specificity. In particular, this thesis presents one of the first attempts to algorithmically detect double minute chromosomes, which are complex rearrangements that are present in many cancers.
Chapter 1

Introduction

1.1 Motivation

Detecting genomic abnormalities is an important problem in bioinformatics with many applications. These abnormalities may include single nucleotide polymorphisms (SNPs), small insertion/deletion polymorphisms (indels), or large structural variants (SV) that are typically greater than 1000 base pairs (bp) in size. Large scale structural variants are prevalent in the human genome, and are ubiquitous amongst populations [1, 2]. These variants are typically innocuous; this is especially true of germline variants, which are inherited from our parents. However, somatically-acquired variants can be harmful, as they may induce disease-causing changes in the genome [3]. Several types of cancer are highly associated with the presence of well-known structural variants such as the Philadelphia chromosome, which is a genomic abnormality that is present in 95% of patients with chronic myelogenous leukemia [4].

Given that SVs may be associated with diseases, it is desirable to not only locate them, but to determine their exact nature. Is the variant a deletion? A translocation? Does it occur in or near a gene that is relevant to preventing a certain disease? These are very important questions but they represent only a small sample of questions that can can be addressed by locating and classifying SVs. Furthermore, this information can assist medical researchers in developing drugs that target certain
diseases, and it can assist in determining the prognosis of an affected patient. There are also implications in the role of SVs and the diagnosis of certain diseases [5].

There are several challenging problems with SV detection. First, laboratory based methods (such as florescent in situ hybridization) are typically low resolution and are inefficient. This problem has served as the primary motivation for not only SV detection using computational approaches, but for the entire field of bioinformatics. The second problem is that even though there are many programs to detect structural variants, many of them do not attempt to detect variants at base pair level (i.e. determining the exact base pairs where the variation starts and ends). Lastly, SVs may be highly complex, involving several exchanges of genomic material amongst different chromosomes (homologous and non-homologous); there is a dearth of published algorithms that attempt to detect and characterize complex variants. There are many more challenges in SV detection than those previously listed, but these are the major questions. Regarding the first problem, several platforms have been developed that facilitate computational analysis of structural variants. One of these is the array comparative genomic hybridization (aCGH) platform, which is designed to assess DNA copy number gains and losses in the genome [6]. Newer platforms have presented next-generation sequencing, which performs DNA sequencing in parallel much cheaper and faster than traditional Sanger sequencing [7, 8]. The aCGH and NGS platforms are discussed in greater detail in Chapter 2. Given the importance of SV detection and the availability of platforms that make detection possible through computational means, it is crucial to develop algorithms that can begin to address the aforementioned challenges with SV detection and classification.
1.2 Objective

This thesis presents several algorithms for detecting structural variation using the aCGH and NGS platforms. We present a very fast algorithm called “CGH-Triangulator” that uses the aCGH platform to segment the genome into regions of distinct copy number [9]. This algorithm creates a set of triangles from the log-2 ratio data points, and it makes predictions based on the spatial characteristics of the resulting triangles. Furthermore, the running time of this algorithm is linear in the number of DNA probes in the input. In addition to CGH-Triangulator, this thesis also presents algorithms for finding and classifying structural variants using NGS datasets. Specifically, we propose an algorithm to classify interchromosomal SVs (called “Bellerophon”), including reciprocal and non-reciprocal translocations, and interchromosomal insertions [10]. This algorithm uses both paired reads and soft-clipped reads to make predictions at base-pair level (Chapter 2). Also, we use this hybrid approach to detect and classify breakpoints that could be caused by deletions, inversions, and tandem repeats. We then present a method that integrates SV breakpoint predictions with copy number segmentation results to predict an oncogenic complex variant known as a double minute chromosome.

Through the aforementioned methods, we are seeking to address the following research questions: 1) for aCGH datasets with low to moderate levels of noise, can they be analyzed substantially faster and more effectively than noisier datasets? ¹ 2) For SVs predicted using NGS, can the combined use of paired-end and split-read

¹We define “effectiveness” as the sensitivity and specificity of change point predictions.
signals lead to increased robustness in SV predictions in the presence of low sequence coverage, while still accurately predicting precise SV breakpoints? 3) Can combining copy number profile with structural variant predictions provide an effective method for identifying double minute chromosomes in the human genome?

1.3 Contribution

The CGH-Triangulator algorithm provides one of the first insights into aCGH change point detection by taking advantage of the spatial characteristics of the log-2 ratio data points. Furthermore, the method is substantially faster and had greater predictive power than a popular method for change point detection called Circular Binary Segmentation (CBS) [11]. Regarding the SV prediction algorithms, they provide one of the first insights into SV detection by combining paired-end signals and split-read signals. By combining the signatures, we found that our sensitivity in predicting variants was robust to decreasing levels of sequence coverage, especially when compared to CREST, a state-of-the-art method that predicts SVs by only considering split reads (i.e. soft-clipped reads). It also had better specificity than the paired end methods, and its breakpoint prediction accuracy was on par with CREST, which can also predict precise breakpoints. Lastly, there are no known algorithms that exist for predicting the existence of double minute chromosomes. This thesis provides the first attempt to address this problem. Our algorithm accurately predicted and reconstructed a simulated double minute chromosome that was derived from a real, experimentally validated variant that was presented in a previous study.
1.4 Organization of thesis

Chapter 2 provides an overview of the aCGH and NGS platforms. It then discusses different types of genomic structural variants and the signals that are used to detect them in aCGH and NGS data. It then discusses how structural variants can make genomic changes that can leave an individual susceptible to the onset of diseases. Chapter 3 provides a review of literature that is relevant to SV detection using NGS data. In Chapter 4, we describe our algorithms in detail and provide the experimental design and results for each method. In Chapter 5, we conclude the thesis and address possible directions for future work.
Chapter 2

Background

2.1 The array CGH platform

Array comparative hybridization (aCGH) is a microarray-based platform that allows for genome-wide screening of genomic copy number changes [6]. In comparison to laboratory methods like SKY karyotyping, the aCGH platform allows for higher resolution analyses of copy number variation. In this platform, DNA probes belonging to a test and a reference sample are fluorescently tagged. The fluorescent intensity of each probe is proportional to the abundance of the probes. This platform is comparative because for each probe, the log base-2 ratio of the fluorescent intensities are measured for the test and a reference samples. Fig. 2.1 illustrates a sample output from an aCGH experiment. Each datapoint is the log-2 ratio of probe intensities. If there are DNA copy number differences between a test and a reference sample, then we can use this platform to see where the copy number change occurs. In most aCGH experiments, the test sample is our sample of interest. For example, it can be derived from an individual affected with some disease, or it could be a tumor sample. The reference sample is assumed to be normal, i.e. with no status of interest.

The aCGH platform provides copy number information for a test vs. reference sample, but computational methods are needed to further analyze these datasets.
First, intensity ratios are a random variable, and not a fixed value, as shown in Fig. 2.1. Secondly, there might be a large amount of aCGH data to analyze, so it may be cumbersome to manually identify segments of constant copy number change. Algorithms to analyze aCGH data typically address one or both of the following problems: 1) find all locations where the distribution of the log-2 ratios changes, and 2) given the output from an aCGH experiment, “smooth” the data by removing extra variation in log ratio values (i.e. reduce the noise). For methods that only address the second problem, they are often used to preprocess the aCGH data before sending it to an algorithm that either estimates copy number or predicts the location of change points [12]. Because there may be a large amount of error in log intensity measurement, these methods are also very important because it is difficult to learn information from aCGH data if it contains a copious amount of noise. The first problem is a hallmark of many methods for aCGH analysis. These methods can be further subdivided into the following categories: a) methods that attempt to estimate
the DNA copy number of a particular segment, and b) methods that find the change points in the data, or the exact locations where the distribution changes. Methods to solve problem “a” by default also solve problem “b”. For some experiments, it may be suitable to simply find the locations where the copy number state changes. These experiments can be addressed by algorithms that address problem “b”.

2.2 Next-generation sequencing platforms

Although the aCGH platform can provide a wealth of information about the copy number profile of a genome, it suffers from two key disadvantages. First, it has limited resolution to capture precise copy number variant breakpoints. Secondly, there are a class of structural variants that are copy-neutral, i.e. their presence does not alter the copy number state of the genome. Such variants will not be detected by the aCGH platform. These problems can be alleviated by using next-generation sequencing (NGS) platforms. NGS platforms are revolutionary in that they allow for the parallel sequencing of genomes much cheaper and faster than traditional methods for sequencing (i.e. Sanger sequencing) [7]. NGS has applications in the areas of SV detection, gene expression analysis, and many other applications. In addition to structural variation detection, NGS technology has also been applied to various domains, including ChIP-seq, RNA-seq, and bisulfite sequencing [13–15]. In the context of SV detection, NGS platforms analyze the genome at base-pair level, so it has higher resolution than the aCGH platform to call variant breakpoints. Also, NGS platforms can not only find copy number variants, but they can also find copy neutral variants. These variants include reciprocal translocations, inversions, and simple
insertions. Different types of structural variants are discussed in Section 2.3.

Illumina, Roche 454 Life Sciences, and Applied Biosystems (ABI) have developed technologies that are commonly used in high throughput sequencing experiments [16]. Illumina sequencing uses a technique known as sequencing by synthesis. In a typical Illumina experiment, the generated fragment lengths are usually around 200-300 bp in length. Current Illumina technologies can produce maximum read lengths of 100-150 bp [17]. Applied Biosciences provides sequencing technologies that are based on the technique of sequencing by oligonucleotide ligation and detection, or SOLiD. ABI platforms can produce reads that are a maximum of 75 bp in length, while fragment sizes are similar to those of Illumina. Roche sequencing platforms employ the pyrosequencing technique. This method produces read lengths that are substantially longer than that of the other technologies: around 700 bp. The Roche platforms also perform sequencing faster than the other technologies [16].

2.2.1 General steps in NGS sequencing experiments

Although there are different NGS platforms with different processes for performing sequencing, they share the same high level steps. First, the genome to be sequenced is isolated and randomly sheared into fragments. For paired-end experiments, the fragments are sequenced from both ends. This step produces sequence reads. The length of the sequence reads may vary depending on the experiment and the platform, but for many experiments, they are typically between 50-100 bp, and the length of the sheared fragments, or inserts, are of roughly known length. The insert sizes can be viewed as a random variable, as the sizes are not always exactly
their intended length. In addition to paired sequence reads, NGS platforms can also create single sequenced reads as well. The process for creating single reads is similar to the process for creating paired reads, except that for single reads, only one end of each fragment is sequenced. Figs. 2.2 through 2.5 depict the formation of paired and single reads from typical NGS experiments.

Figure 2.2: NGS sequencing step 1: isolate DNA.

Figure 2.3: NGS sequencing step 2: create fragments through random shearing.
2.3 Structural variation

As previously stated, there are several types of genomic variation, but large structural variants tend to be greater than 1000 bp in size. These variants may be located in regions of the genome that are functionally important in the prevention of diseases. SVs have been linked to disorders such as cancer, infertility, and autism [1, 2, 18, 19]. The next subsection reviews five basic types of structural variants.
2.3.1 Basic types

2.3.1.1 Deletions

A genomic deletion occurs when some contiguous portion of the genome has been lost due to an external factor. Fig. 2.6 depicts a deletion of a segment from a chromosome $i$. If a deletion occurs in or near a particular gene, it may have a profound impact on the health of an individual, especially if the deletion is homozygous (i.e. if it occurs on both haplotypes). In the case of a homozygous deletion, the genome is missing both copies of the gene, so the individual may be susceptible to the conditions that may arise due to this kind of variant. In the context of cancer genomics, a deletion in a tumor-suppressing gene may increase the susceptibility of an individual to cancer. An example of such a deletion are those present in the TP53 gene. This gene codes for the p53 protein, which is known as the “guardian” of the genome [20]. Studies have shown that deletions in this gene are highly associated with the presence of chronic lymphocytic leukemia (CLL) [21]. Deletions from this tumor suppressor hinders the translation of p53, which naturally leads to tumor onset.

In addition to affecting tumor suppressing genes, deletions may also cause fusion genes [22]. As their name implies, fusion genes are created by the adjoining of two previously separate genes. In the case of more than two genes being fused, the phenomenon is known as a poly-gene fusion. Fig. 2.7 illustrates how deletions may create fusion genes. These genes may be of clinical importance because they may code for a protein that is highly oncogenic [23, 24].
2.3.1.2 Insertions

Insertions are caused by the addition of some nucleotide base pairs into a DNA sequence. These inserted sequences may come from another (non-homologous) chromosome, as in the case of *interchromosomal insertions*, or in the case of *transposon insertions*, it could come from another segment of the same chromosome [25, 26]. If the variant is a *simple insertion*, the inserted segment may be a novel sequence of
unknown origin [27]. Fig. 2.8 depicts an insertion event. As with deletions, insertions can disrupt gene functions. If foreign DNA is inserted into an exon, then the gene may not properly translate. Thus, a tumor-suppressing gene containing an insertion could lose its function. Furthermore, the inserted DNA could contain functional parts of a gene, so such an insertion could also create fusion genes. This especially true for interchromosomal insertions and transposon insertions. Fig. 2.9 depicts a fusion gene caused by an interchromosomal insertion. For interchromosomal insertions, the inserted segment may be inverted with respect to the centromere. In this case, the event is an inverted insertion, else the event is a direct insertion [26].

Figure 2.8: Structural variation: insertions.

Figure 2.9: A gene fusion caused by an interchromosomal insertion. Gene A and gene B are located on separate, non-homologous chromosomes. A segment from gene B has been inserted into a region of gene A, thus creating a fusion.
2.3.1.3 Inversions

Inversions are another kind of structural variant. An inversion occurs when a given genomic segment changes its orientation with respect to the centromere. Fig. 2.10 illustrates this. Unlike deletions, inversions are *copy neutral* variants, as their presence does not alter the number of copies of the affected genome region. However, inversions may still adversely affect the health of an individual. These variants have been found to be associated with other SVs, such as deletions. For example, one study found that some individuals affected with Williams-Beuren syndrome contain a 1.5 Mb microdeletion at 7q11, and this segment was inverted in at least one of their parents [28]. Furthermore, inversions may create fusion genes, as depicted in Fig. 2.11 [29].

![Chromosome i](image1)

![Chromosome i with inversion](image2)

Figure 2.10: Structural variation: inversions. The colored circles represent DNA sequences. When an inversion occurs, this segment will change its orientation.
Figure 2.11: Gene fusion caused by an inversion. The segment within the green box is inverted, causing the fusions depicted in the bottom image.

2.3.1.4 Tandem repeats

As their name implies, tandem repeats are caused by a replication of a series of nucleotides, where the replicates are directly adjacent to one another. Fig. 2.12 illustrates a tandem repeat of a genomic segment. Although only two copies were produced in the figure, any number of copies can result from tandem replication. If a tandem repeat encompasses a gene, then it could lead to increased expression of the gene since multiple copies are present. This is particularly problematic if the gene is an oncogene, which is a gene whose overexpression may lead to cancer onset. An example of such an oncogene is the RPS6KB1 gene, which codes for a protein that increases the rate of cell growth. Overexpression of this gene is linked to breast cancer [30]. In general, the overexpression of genes that contribute to cell division may leave an individual susceptible to tumorigenesis.
2.3.1.5 Translocations

A translocation occurs when two non-homologous chromosomes break apart and fuse to each other. If the two chromosomes exchange arms during the fusion, then the translocation is balanced, or reciprocal, else, the translocation is unbalanced or non-reciprocal. These two types of translocations are illustrated in Figs. 2.13 and 2.14. Translocations can play a major role in the onset of diseases. Like deletions, a translocation can lead to a loss of genomic material if the variant is non-reciprocal. Also, gene fusions may naturally occur as a result of a translocation [24]. As stated before, such fusions may be highly oncogenic. An example of a gene fusion caused by a balanced translocation is the Philadelphia chromosome [4]. This variant results from a reciprocal translocation between chromosomes 9 and 22. As a result, the BCR and ABL genes are fused together. This is a lethal gene fusion, as 95% of patients with chronic myelogenous leukemia have the Philadelphia chromosome.
2.3.2 Complex structural variants

Complex structural variants generally refer to SVs that involve genomic material from more than two non-homologous chromosomes, or that involve intrachromosomal segments that are highly rearranged. Like basic SV types, the presence of a complex variant may be associated with certain cancers.
2.3.2.1 Balanced closed chain rearrangements

Berger and colleagues presented a complex genomic rearrangement known as a balanced closed chain rearrangement (BCCR) [31]. This phenomenon is presented in Fig. 2.15. This complex event occurs when more than two chromosomes engage in a balanced exchange of breakpoint arms. It can be viewed as a generalization of a reciprocal translocation. In their study, Berger et al. discovered two of these complex variants. The first BCCR created a TMPRSS2-ERG gene fusion, which is a fusion that is highly associated with prostate cancer. The other BCCR was much larger, and it contained breakpoints in or near genes that are cancer related, such as ABL1, TBK1, MAP2K4, and TP53. They indicated that the presence of BCCRs could mean that different genomic regions may be co-localized prior to rearrangement. Given the information learned in Berger’s study, it is likely that BCCRs are important markers for cancer, especially if they create many fusion genes as seen in their study.

2.3.2.2 Chromothripsis and double minute chromosomes

A recent study by Stephens et al. presented a newly discovered phenomenon known as chromothripsis, which literally translates as “genome shattering” [32]. Until recently, many researchers have assumed that changes to the genome during cancer development have been gradual. Many now believe that these rearrangements may be the result of a single catastrophic event. As its name implies, this phenomenon is the result of the genome shattering haphazardly, primarily during cancer development. As a result, the genome will attempt to repair itself, but the repairs will likely be erroneous. The chromothripsis phenomenon is presented in Fig. 2.16, and an example
Figure 2.15: Complex SV: balanced closed chain rearrangement. There is a perfect balanced exchange of arms amongst the three chromosomes.

of an erroneous repair is shown in Fig. 2.17. Also shown in Fig. 2.17 is the formation of a double minute chromosome, which is a small, circular segment of extrachromosomal material that may form as a result of chromothripsis, though they are generally formed as a result of gene amplification during tumorigenesis [33, 34]. Chromothripsis is not a necessary phenomenon for the existence of double minute chromosomes.

Chromothripsis and double minute chromosomes are complex structural variants that are clearly associated with cancer. During chromothripsis, the haphazard shattering and repair can lead to the formation of oncogenic fusion genes. In their study, Stephens et al. note that chromothripsis is seen in at least 2%-3% of all cancers, and that it is present in approximately 25% of bone cancers. Regarding double
Figure 2.16: Complex SV: chromothripsis. During cancer development, the genome may shatter in a single catastrophic event.

Figure 2.17: Erroneous repair of a genomic region affected by chromothripsis. The segments comprising the genome subset may be highly rearranged after the event. It is even possible to see the formation of double minutes.

minutes, they harbor the lethal combination of 1) being highly amplified and 2) containing oncogenes. They have been found in a wide range of cancers, and in one study, they were even found to contain genes that enable drug resistance of certain carcinoma cells [35]. The amplification of the Fig. 2.17 double minute is provided in Fig. 2.18.
2.4 Detecting structural variants using NGS

Given the potential impact of SVs on the human genome, it is important for researchers to have available efficient and accurate methods to locate and classify them. Using next-generation sequencing to find structural variants generally requires the following steps. First, a test (e.g. diseased) genome of interest is sequenced using some NGS platform. Secondly, the resulting sequence reads are aligned to a reference genome using a short read alignment program. Bowtie, BWA, and SOAP are popular examples of programs used for this purpose [36–38]. If structural variation has occurred, then the reads flanking the SV breakpoint will have a discordant alignment when mapped back to the reference genome. An alignment is discordant if either 1) the read pair aligns with mapped distance between the reads that is greater than expected, or less than expected, 2) the orientation of the mapped read pairs is different than before alignment, or 3) if the two reads align to different chromosomes. If none
of these conditions are true, then the read alignments are consistent with the cor-
responding fragment’s characteristics during sequencing. In this case, we say that the 
alignment is *concordant*. Once the sequence reads have been aligned, an SV detection 
algorithm will search for discordant alignments, because they indicate a possible SV. 
Fig. 2.19 provides a visual description of SVs and the discordant alignments that 
support them. Some NGS-based programs may only observe read depth to make its 
SV calls, but such methods cannot detect copy neutral variants like inversions and 
translocations. Furthermore, some methods may use single-read or “split-read” ap-
proaches to SV predictions, but the resulting mapping signatures are similar to that 
of paired reads.

![Diagram of Structural Variants](image)

Figure 2.19: Structural variants and the discordant alignments that support them. 
The colored circles for the inversion case represent DNA sequences in the affected 
segment. Most NGS-based algorithms predict structural variants by searching for 
these anomalously mapped read pairs. The “D” represents the donor genome, and 
the “R” represents the reference genome.

To detect more complex variants, it is necessary to have an algorithm that can
somehow associate together discordant alignments of different types. For example, double minute chromosomes and chromothripsis can produce genome products that simultaneously resemble deletions, translocations, inversions, and tandem repeats. A BCCR may simultaneously resemble an interchromosomal translocation or an intra-chromosomal translocation (i.e. like a transposon). A method that only finds basic types of SVs will find each of these signals separately, but it cannot consolidate them into a single complex variant prediction. Chapter 3 references a recent study that presents an algorithm to detect BCCR and chromothripsis.
Chapter 3

Literature Review

Several algorithms have been published that address the problem of finding structural variants using next-generation sequencing data. These methods may differ in the types of structural variants they locate and classify. They may also differ in the types of NGS-produced signals they consider when making predictions, although most methods search for the paired-end patterns illustrated in Fig. 2.19. Among the methods that only consider the paired-end signal are BreakDancerMax, GASV, and SVDetect. SVDetect divides the genome into overlapping windows and predicts structural variants by assessing windows that are linked by anomalously mapped paired reads [39]. BreakdancerMax identifies potential variants by locating regions that contain more abnormally mapped read pairs than is expected. It then uses a Poisson model to calculate a confidence score for each candidate variant [40]. GASV presents a geometric approach to SV detection. This algorithm identifies regions of breakpoint uncertainty and constructs polygons representing these regions. It then finds structural variants by computing the number of intersecting polygons for a given region [41]. In addition to the aforementioned programs, our lab has developed a method called “SVMiner” that can detect deletions and inversions in NGS data [42]. This method creates a feature space for potential deletions and inversions, and it makes its final predictions by applying a model-based clustering algorithm on the
data. Unlike many methods to detect SVs, SVMiner can genotype deletion variants.

Some SV detection algorithms make predictions by only using the information produced from mapping single reads to the reference. The advantage of single-read (or “split-read”) methods is that they can detect variants at base pair level, which paired-end methods cannot do. Detecting SVs at base-pair level is important, because knowing the precise breakpoints of variants would further assist researchers in locating SVs as therapeutic targets. An example of a split-read algorithm is Pindel, which can detect large deletions and medium-sized insertions [43]. It employs a pattern growth approach to detect these variants at base-pair level. A more recent split-read method is CREST. This method uses reads that contain soft-clipped alignments. Soft-clipped reads contain a contiguous match to the reference, but another contiguous part of the read may be mappable elsewhere [44]. CREST uses these soft-clipped reads to find putative variant breakpoints by performing DNA assembly on the clipped subreads, and then by subsequently mapping the assembled contig back to the reference genome.

Two recently published methods called Delly and RetroSeq combine the use of the paired-end signal and the split-read signal to make SV predictions. Combining these signals is advantageous since 1) split-read methods may be less sensitive in low-coverage regions, and 2) paired-end methods cannot make precise breakpoint calls. The Delly algorithm uses the combined approach to find deletions, tandem repeats, inversions, and translocations [45]. It uses a graph-based approach to first find groups of discordant alignments that could denote a possible SV. It then applies a dynamic programming-based routine to align single reads whose mates were aligned to the same region as the discordant mate pairs. This allows predictions to be made at
base-pair level. The RetroSeq algorithm finds transposons by using the combined approach [46]. It first searches for discordant mate pairs and it then assigns them to an appropriate transposable element class. In its second phase, it then uses soft-clipped reads to refine breakpoints.

The aforementioned methods can likely be used to find breakpoints in complex structural variants, but they cannot predict the existence of the complex variant itself. Overall, there are a dearth of methods that can find complex structural variants. However, the nFUSE algorithm can detect chromothripsis and balanced closed chain rearrangements in NGS datasets [47]. This method incorporates genome sequencing data with cDNA sequencing data (i.e. RNA-seq) to find complex rearrangements that underlie fusion transcripts. The core of the method finds complex SVs by searching for a shortest alternating path in a breakpoint graph [48]. In general, detecting complex SVs requires a method that can examine the properties of distinct SV breakpoint predictions and consolidate them into a single prediction. In the next chapter, we describe a method to predict double minute chromosomes, a complex event that nFUSE cannot find directly.
Chapter 4

Algorithms

4.1 Detecting copy number change points in aCGH data using log ratio triangulation

4.1.1 Motivation

As previously stated, the aCGH platform is an effective medium for assessing the copy number profile of the genome or a subset of the genome. Changes in copy number are seen in many cancers, so it is important to devise methods that can assist in revealing the copy number profile of the genome, especially the genomes of patients [49, 50]. Such information would assist a medical researcher in gaining important clinical information about a patient, such as the disease stage and classification of tumor cells. However, the noisy nature of aCGH data obfuscates the true copy number signal that would otherwise be present. To address this, we have developed an algorithm called CGH-Triangulator, which segments the log ratio values into overlapping triangles and subsequently takes advantage of the spatial characteristics of the triangles to make predictions about the location the copy number change points. This is a fast algorithm that runs in time that is linear to the number of data points in the input.
4.1.2 Methods

4.1.2.1 Log-2 ratio triangulation

The idea of the triangulation\(^1\) method is to exploit the behavior of log-2 ratio values that are often seen at change point locations. By forming triangles from every three consecutive ratio values on the chromosome, we seek to computationally compare the triangles formed in segments of constant copy number to triangles formed at the segment boundaries. To distinguish between these two types of triangles, we define two score functions \(S\) and \(G\) that assign values to each triangle. The \(S\)-score for a triangle \(\tau\) is defined by the following equation:

\[
S(\tau) := F^{cHT^\circ}.
\]  

\(F\) is the value in radians of the largest interior angle of \(\tau\), \(T\) is the triangle’s tilt off the x-axis in degrees, \(H\) is the height of \(\tau\), and \(c\) is a user specified parameter whose effects are explored in Section 4.1.3.\(^2\) Fig. 4.1 provides an illustrative example of the \(T, F,\) and \(H\) dimensions and how they are determined from log-2 ratio values.

After calculating the value in (1), \(S(\tau)\) is then given as input to the following function \(G(S)\), which we refer to as the \(G\)-score.

\[
G(S(\tau)) := \arctan(\gamma(S(\tau) - d)) .
\]  

\(^1\)Note that we define triangulation as the task of forming a triangle from 3 points in two dimensional space.

\(^2\)We specify the tilt dimension in degrees so that it produces a larger value to aid in separating triangles at breakpoints from triangles in constant copy number segments.
Figure 4.1: Illustrative example of $T,F$, and $H$ for a given triangle. The points $x_1$, $x_2$, and $x_3$ are log-2 ratio values in the aCGH data, and triangles are created for every three consecutive log-2 ratio values.

In (2), the variable $\gamma$ is the estimated noise dispersion from the segment means of all log-2 ratios for the given input set of clones. The variable $d$ is a user-specified constant that specifies the x-intercept of the arctangent function. The effects of this parameter are studied further in Section 4.1.3. The algorithm to calculate $\gamma$ is provided in subsection 4.1.2.3.

The idea of the score functions is not only to emphasize the importance of the $T,F$, and $H$ dimensions, but to exploit the properties of triangles that are formed at the breakpoint locations. These triangles tend to have larger heights, greater tilt, and larger maximum interior angles than other triangles. Since these triangles are formed from log-2 ratio values in the data, we are essentially exploiting the noisiness of the data and projecting the related properties onto properties of the associated triangles. Equations (1) and (2) also allow our method to numerically separate triangles seen at copy number transition points from triangles obtained from clones with the same copy number. For our purposes, we want to place smaller triangles with low $S$-score values in the lower asymptote region of the $G$-score curve, while triangles with higher
$S$-scores should be placed in the upper asymptote region. We chose a sigmoid as our $G$ function because at a certain point, we do not want scores to increase because a large score value for a single triangle would hinder our algorithm’s ability to locate change points indexed by smaller triangles.

### 4.1.2.2 The algorithm

The Triangulator algorithm is shown in Fig. 4.2. The algorithm first reads in an array of clones $A$ ordered by genomic position, and each element in $A$ contains information about the clone’s genomic location and it’s log-2 ratio. The genomic locations for the clones are temporarily stored to an array $P$ so that we can “normalize” the distances between the clones to ensure that all triangles formed are of the same width. The original genomic position values are indexed by the value of the new position, so at termination, the original positions are used in estimating change point locations.

For every three consecutive clones, the log-2 ratios for the clones are used to create a triangle. This step is shown on line 8. The $\text{Triangulate}$ function simply creates a triangle from the three clones given as input where the first, second, and third parameters correspond to the first, second, and third points in the triangle, respectively ordered by their position on the x-axis. The $G$-score of each triangle is stored to the array $M$, and on line 11, the actual triangles are stored to the array $T$. Moreover, each triangle is indexed by its first point on the x-axis, as shown on line 9. This indexing facilitates the process of associating triangles with genomic locations.

Lines 14 through 17 identify an initial set of triangles that are candidates for
**input:** An array $A$ of $n$ DNA clones along a chromosome.

**output:** A set $C$ of genomic locations (in kilobases) where the locations are predicted copy number break points.

```
1 $M \leftarrow \emptyset$;
2 $P \leftarrow \emptyset$;
3 $T \leftarrow \emptyset$;
4 for $h \leftarrow 1$ to $n - 2$ do
5   $P[h] \leftarrow A_{Genomic\ Position}[h]$;
6   $A_{Genomic\ Position}[h] \leftarrow h$;
7 for $i \leftarrow 1$ to $n - 2$ do
8   $\tau \leftarrow Triangulate(A[i], A[i+1], A[i+2])$;
9   $\tau_{index} \leftarrow i$;
10  $M[i] \leftarrow G(S(\tau))$;
11  $T[i] \leftarrow \tau$;
12 $T_a \leftarrow \emptyset$;
13 numAberrant $\leftarrow 1$;
14 for $j \leftarrow 1$ to $n - 2$ do
15   if $M[j] > \mu(M) + \sigma(M)$ or $M[j] < \mu(M) - \sigma(M)$ then
16     $T_a[numAberrant] \leftarrow T[j]$;
17     numAberrant $\leftarrow$ numAberrant + 1;
18 for $k \leftarrow 1$ to length($T_a$) do
19   $R \leftarrow \emptyset$;
20   $e \leftarrow T_{a_{index}}[k]$;
21   if $e > 0$ then
22     $e \leftarrow T_{a_{index}}[k] - 1$;
23     $a \leftarrow T_{point1}[e]$;
24     $a_{Genomic\ Position} \leftarrow T_{a_{index}}[k] + 1$;
25     for $p \leftarrow T_{a_{index}}[k] + 1$ to $T_{a_{index}}[k + 1] - 1$ do
26       $b \leftarrow T_{point2}[p]$;
27       $c \leftarrow T_{point3}[p]$;
28       $\tau \leftarrow Triangulate(a, b, c)$;
29       $R \leftarrow R \cup G(S(\tau))$;
30     $a_{Genomic\ Position} \leftarrow a_{Genomic\ Position} + 1$;
31   if $G(S(T_a[k])) \leq \mu(R) + \sigma(R)$ and $G(S(T_a[k])) \geq \mu(R) - \sigma(R)$ then
32     $pos \leftarrow T_{a_{index}}[k]$;
33     $C \leftarrow C \cup P[pos]$;
34 return $C$;
```

Figure 4.2: CGH-Triangulator algorithm
breakpoint locations. For each triangle in the array, its $G$-score is compared to the mean $G$-score of all triangles in $T$. If the $G$-score for a given triangle is not within one standard deviation of the mean of all $G$-scores, then the triangle is a candidate for a true breakpoint. All such triangles are stored to the array $T_a$ in increasing order by genomic position.

Lines 18 through 33 are the part of the algorithm that determines if the aberrant triangles in $T_a$ are indexed at true change points or if they are simply indexed at noisy points. The index of the $k$th aberrant triangle is stored to $e$ at line 20. The inner for-loop at line 25 loops between the index values for consecutive aberrant triangles. In this step, the segment of the chromosome between the indexed aberrant triangles is analyzed by triangulating the 1st point of the triangle at $T[e]$ with the 2nd and 3rd points of the triangle indexed by $p$. The score value from the resulting triangle is stored to the set $R$ and this process is repeated for all points in the current segment. Referring to lines 31 through 33, if the aberrant triangle $T_a[k]$ is within 1 standard deviation of the mean of the score values in $R$, the clone indexed by $T_a[k]$ is identified as a change point and is added to the set $C$.

4.1.2.3 Algorithm to calculate $\gamma$

We calculate $\gamma$ by first performing median filtering on the input log-2 ratios (with a window size of 20) and for each original ratio value, we take the difference between that value and its corresponding filtered value. We then sum the differences (i.e., their absolute values) of all clones to their filtered values and we divide the resulting sum by the number of clones in the input set. The result is a value that
serves as a measure of noise level for the input data and such a value is important because array CGH datasets vary in the levels of noise they contain. The $\gamma$ parameter also affects the curvature of the sigmoid and the rate at which the upper and lower asymptotes are reached. A larger $\gamma$ value will give a steeper curve and will have a larger rate of change to reach the lower and upper asymptotes. This effect is desirable because $\gamma$ varies directly with noise, and noise varies directly with $S$-score values. Thus, a noisy dataset will have triangles with larger $S$-scores, and we want our sigmoid $G$ function to account for these larger values.

4.1.2.4 Proof of time complexity

Referring to Fig. 4.2, lines 4-11 of the algorithm are clearly linear time. The mean and standard deviation of the triangle $G$-scores can be calculated in linear time using an online algorithm such as that provided by Donald Knuth [51]. Thus, lines 14-17 are also executed in linear time. Lines 18-30 present nested loops, and it would appear that each loop is bounded by $O(n)$ in the worst case, which would yield a total running time of $O(n^2)$. The outer loop iterates on the number of abberant triangles, which is bounded by $n/3$, and the inner loop iterates on the number of triangles between aberrant triangles, which is also bounded by $n/3$. However, we can derive much tighter bounds by noting that at the worst case, the algorithm will iterate once over all data points that are in between all aberrant triangles. Since the

---

3We do not refer to this value as the estimated noise standard deviation because we do not take the mean of the values in the window, nor do we square the difference between the observed and expected values. We observed that such a method poorly estimates the true noise standard deviation.
set of data points between all aberrant triangles is simply the input set itself, we have the following inequality:

\[ \sum_{a=1}^{T_a-1} (N_{a,a+1}) \leq N \tag{4.3} \]

\( N \) is the total number of input data points, \( T_a \) is the number of aberrant triangles, and \( N_{a,a+1} \) is the number of data points between aberrant triangles \( a \) and \( a + 1 \). Thus, lines 18-30 also run in \( O(n) \) time. The remainder of the algorithm is a simple if-statement. Also, the algorithm to calculate \( \gamma \) adds no asymptotic overhead since it can be calculated in \( O(n) \) as a preprocessing step. Therefore, the worst case running time of this algorithm is \( O(n) \), which is an improvement over many earlier methods to address this problem.

### 4.1.3 Experiments

#### 4.1.3.1 Study on the Coriell Cell Line

We first tested our algorithm on an array CGH dataset from the Coriell Institute that was first published in [52]. This dataset is annotated with true copy number state information which allowed for quantitative evaluation of the accuracy of our segmentation method. Moreover, it is important to have this information for real data so that we can make predictions as to the performance of our method on real datasets.

To measure accuracy of predicted breakpoints, we measured the sensitivity (SE) as the proportion of true breakpoints identified to all real breakpoints, and we also measured the false discovery rate (FDR) as the proportion of falsely-predicted
breakpoints to all predicted breakpoints. We also ran a Matlab implementation of the CBS algorithm and we compared the performance of the two programs in regards to accuracy and computation time. For the user parameters $c$ and $d$, referenced in Section 3, we ran the experiment for $c = \{2, 3\}$ and for each value of $c$, we chose $d = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$. For $c > 3$, we noticed that the method returned a high number of false positives, and for $c = 1$, a low number number of true positives were returned. Thus, we focused our attention only on two values of $c$.

4.1.3.2 Results: Coriell Cell Line

The results from this study are provided in Tables 1 and 2, and the results of the CBS algorithm on this dataset are given in Table 3. The tables suggest that the results are generally robust against different choices of parameters for $c$ and $d$, though the proportion of false positives is higher for the $c = 3$ experiment. This is especially true for lower values of $d$ when $c = 3$ because the $S$-score values are higher. This causes the $G$-score function to lose effectiveness in separating normal and aberrant triangles. We note, however, that the number of false positives decreased as $d$ grew larger. In both sets of results, we observed that as $d$ grows larger, the FDR will decrease before the method begins to lose sensitivity. Although not reported in Table 2, we observed that for $c = 3$, the value of $d$ can increase to approximately 10000 before the method begins to lose sensitivity.\footnote{For $c = 3$ and $d = 10000$, SE = 0.86, FDR = 0.22.} For $c = 2$, the drop in sensitivity occurs much sooner.
As suggested in [53], we used a variable $w$ to represent the maximum allowed localization error of clones, so a breakpoint was correctly identified if it was within $w$ clones of the true breakpoint. For both values of $c$, the triangulation method generally outperforms CBS with regard to FDR, while SE is never less than 0.80 for $w = 1$. For $w = 0$, the Triangulator algorithm is clearly more precise than the CBS algorithm on this dataset.

The sensitivity of both algorithms improved when we increased $w$ from 0 to 1. This is mainly because the data does not have sharp breakpoints in some cases. For our algorithm, reasons for loss of sensitivity include lack of breakpoint “sharpness” and segments that were of length 1. Our method relies on breakpoints that are defined by large, abnormal triangles. But on occasion, the breakpoints are defined by smaller triangles whose points are gradually increasing. There is also a loss of sensitivity with segments of length 1 because the method treats them as single, noisy data points. We observed that CBS will also miss segments of length 1. As previously stated, our reported FDR is generally smaller than that of CBS, which is a desirable feature.

In regards to time, our study confirmed our assumption that the Triangulator algorithm would be much faster than the CBS algorithm. In each case, our algorithm segmented the 15 cell lines in under 11 seconds, while the CBS algorithm needed over 25 minutes to segment these data.

4.1.3.3 Study on simulated dataset

The second part of our study involved the application of our algorithm to synthetic aCGH data. The simulation model used was that of Willenbrock et al. [53].
Table 4.1: For $c = 2$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds.

<table>
<thead>
<tr>
<th>$d$</th>
<th>SE, FDR ($w = 0$)</th>
<th>SE, FDR ($w = 1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.77, 0.58</td>
<td>0.82, 0.55</td>
</tr>
<tr>
<td>200</td>
<td>0.80, 0.375</td>
<td>0.86, 0.32</td>
</tr>
<tr>
<td>300</td>
<td>0.80, 0.31</td>
<td>0.86, 0.25</td>
</tr>
<tr>
<td>400</td>
<td>0.77, 0.32</td>
<td>0.84, 0.26</td>
</tr>
<tr>
<td>500</td>
<td>0.72, 0.36</td>
<td>0.81, 0.26</td>
</tr>
<tr>
<td>600</td>
<td>0.70, 0.34</td>
<td>0.80, 0.25</td>
</tr>
<tr>
<td>700</td>
<td>0.70, 0.30</td>
<td>0.80, 0.20</td>
</tr>
<tr>
<td>800</td>
<td>0.70, 0.32</td>
<td>0.80, 0.24</td>
</tr>
<tr>
<td>900</td>
<td>0.70, 0.26</td>
<td>0.80, 0.16</td>
</tr>
<tr>
<td>1000</td>
<td>0.70, 0.32</td>
<td>0.80, 0.16</td>
</tr>
</tbody>
</table>

Table 4.2: For $c = 3$. Results of Triangulator algorithm on Coriell data. Each test run required approximately 11 seconds.

<table>
<thead>
<tr>
<th>$d$</th>
<th>SE, FDR ($w = 0$)</th>
<th>SE, FDR ($w = 1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.77, 0.80</td>
<td>0.82, 0.55</td>
</tr>
<tr>
<td>200</td>
<td>0.82, 0.71</td>
<td>0.89, 0.68</td>
</tr>
<tr>
<td>300</td>
<td>0.80, 0.65</td>
<td>0.86, 0.65</td>
</tr>
<tr>
<td>400</td>
<td>0.80, 0.60</td>
<td>0.86, 0.57</td>
</tr>
<tr>
<td>500</td>
<td>0.80, 0.59</td>
<td>0.86, 0.55</td>
</tr>
<tr>
<td>600</td>
<td>0.80, 0.57</td>
<td>0.80, 0.54</td>
</tr>
<tr>
<td>700</td>
<td>0.80, 0.51</td>
<td>0.86, 0.46</td>
</tr>
<tr>
<td>800</td>
<td>0.80, 0.49</td>
<td>0.86, 0.44</td>
</tr>
<tr>
<td>900</td>
<td>0.82, 0.47</td>
<td>0.86, 0.44</td>
</tr>
<tr>
<td>1000</td>
<td>0.82, 0.45</td>
<td>0.86, 0.41</td>
</tr>
</tbody>
</table>

Table 4.3: Results of the CBS algorithm on Coriell dataset. Results are consistent with those in the study of Olshen et al.

<table>
<thead>
<tr>
<th>Programs</th>
<th>SE, FDR ($w = 0$)</th>
<th>SE, FDR ($w = 1$)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS</td>
<td>0.600, 0.769</td>
<td>0.900, 0.654</td>
<td>&gt; 25m0s</td>
</tr>
</tbody>
</table>

In this paper, they assume that not all cells in a sampled culture will be affected by copy number changes, but that only a proportion of cells will be diseased. Thus, they
assume that the expected log-2 ratio of a clone is given by the following equation:

\[
\text{ratio} := \log_2[(c \cdot P_t + 2 \cdot (1 - P_t))/2].
\]  

(4.4)

where \(c\) is the copy number state (0,1,2,3,4,5) and \(P_t\) is a proportion of tumorous cells typically seen in biopsies, sampled from a uniform distribution on \([0.3,0.7]\). After adding Gaussian noise sampled from \(\text{Normal}(0, \sigma)\), where \(\sigma\) is sampled from \(\text{Uniform}(0.1,0.2)\), the final value of the clone’s log ratio is determined.

As in the study involving real aCGH data, we measured the sensitivity and false discovery rates of breakpoint prediction accuracy using the simulated data created from the model in [53]. However, we not only wanted to measure the accuracy of predicted breakpoints, but we wanted to determine the level of noise that would cause our algorithm to give undesirable results. We therefore altered the simulation model by assigning different values to the parameters of the uniform distribution used for the \(\sigma\) variable. The \(\sigma\) parameter is the standard deviation for the Gaussian noise added to the log-2 ratios. By restricting the values of these parameters at each step, we determine the suboptimal noise level by measuring sensitivity and false discovery rate of predicted breakpoints.

For this part of the experiment, we measured sensitivity and FDR for parameter \((\sigma)\) values of \([0.0,0.05]\), \([0.0,0.1]\), \([0.05,0.1]\), and \([0.1,0.15]\). For each of these parameter settings, we generated 500 samples with 20 chromosomes, each with 100 clones. This is similar to the experiment performed in [53]. As in the study on the Coriell data, we calculated SE and FDR for \(w = 0.1\). We did not redo this experiment using CBS because the comparison paper [53] had done an extensive study of CBS.
on the simulated data. For this part of the study, we chose values of $c = \{2, 3\}$ and $d = \{100, 200, 300, 400, 500\}$.

### 4.1.3.4 Results: simulated data

The results of this study are given in Tables 4 and 5. As seen in the tables, the method performs well on data with low to moderate noise levels, but loses accuracy as the amount of noise increases. Increasing $w$ from 0 to 1 causes a slight increase in sensitivity, but a sharper decline in the number of false positives. This improvement from $w = 0$ to $w = 1$ is consistent with the results from the Coriell data, in which the results also improved when the offset value increased. It should be noted that the CBS algorithm gave more consistent results even with a noise level higher than those specified in our study (i.e., $(\sigma \in [0.1, 0.2])$). Our method would likely be best suited for microarray experiments in which the researcher can expect low to moderate noise levels. Examples of real data with such noise levels are the Coriell data, that we observed had a noise variance of around 0.1. Furthermore, we used a simulation model that assumed that a sample from a cell line contained only a portion of cells affected by copy number. As a result, the log-2 ratio values calculated for each clone were smaller than they would be if the assumption was that all cells in a culture were affected by copy number changes. Ultimately, if the data is very noisy, then triangles within noisy segments will be very large. Moreover, breakpoints will not be sharply defined by single, oblong triangles, but may instead be defined by multiple triangles that are indistinguishable from other triangles in the data. This lack of sharply defined breakpoints will certainly cause the method to lose sensitivity.
Table 4.4: For $c = 2$. Results of Study on Simulated Data. Each test run required approximately 5m30s.

<table>
<thead>
<tr>
<th>$d$</th>
<th>Noise</th>
<th>SE,FDR ($w=0$)</th>
<th>SE,FDR ($w=1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.0, 0.05</td>
<td>0.80, 0.02</td>
<td>0.82, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.77, 0.07</td>
<td>0.80, 0.03</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.70, 0.17</td>
<td>0.76, 0.09</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.55, 0.72</td>
<td>0.70, 0.64</td>
</tr>
<tr>
<td>200</td>
<td>0.0, 0.05</td>
<td>0.75, 0.03</td>
<td>0.77, 0.006</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.72, 0.07</td>
<td>0.75, 0.02</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.65, 0.13</td>
<td>0.72, 0.05</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.54, 0.64</td>
<td>0.67, 0.54</td>
</tr>
<tr>
<td>300</td>
<td>0.0, 0.05</td>
<td>0.73, 0.03</td>
<td>0.74, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.69, 0.06</td>
<td>0.72, 0.02</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.65, 0.12</td>
<td>0.69, 0.03</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.52, 0.59</td>
<td>0.65, 0.49</td>
</tr>
<tr>
<td>400</td>
<td>0.0, 0.05</td>
<td>0.71, 0.02</td>
<td>0.73, 0.006</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.67, 0.07</td>
<td>0.71, 0.02</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.61, 0.12</td>
<td>0.67, 0.03</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.52, 0.56</td>
<td>0.64, 0.45</td>
</tr>
<tr>
<td>500</td>
<td>0.0, 0.05</td>
<td>0.70, 0.03</td>
<td>0.715, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.65, 0.07</td>
<td>0.69, 0.01</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.60, 0.11</td>
<td>0.66, 0.02</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.51, 0.53</td>
<td>0.63, 0.42</td>
</tr>
</tbody>
</table>

The simulated dataset appears to be more sensitive to choices of $c$ than the Coriell dataset, though for both values of $c$, the method generally has a low FDR. Regarding sensitivity, lower values of $d$ increase the number of true positives, though the number of true positives increases significantly for $c = 3$. For both values of $c$, the proportion of false positives was clearly higher in the Coriell dataset than in the simulated dataset. This difference can be attributed to differences between the two datasets, such as the number of copy number states, noise standard deviation,
Table 4.5: For \( c = 3 \). Results of Study on Simulated Data. Each test run required approximately 5m30s.

<table>
<thead>
<tr>
<th>( d )</th>
<th>Noise</th>
<th>SE,FDR ((w=0))</th>
<th>SE,FDR ((w=1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.0, 0.05</td>
<td>0.91, 0.02</td>
<td>0.92, 0.006</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.86, 0.16</td>
<td>0.90, 0.12</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.78, 0.35</td>
<td>0.86, 0.28</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.56, 0.86</td>
<td>0.76, 0.80</td>
</tr>
<tr>
<td>200</td>
<td>0.0, 0.05</td>
<td>0.88, 0.02</td>
<td>0.90, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.84, 0.12</td>
<td>0.88, 0.08</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.76, 0.26</td>
<td>0.83, 0.18</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.56, 0.86</td>
<td>0.76, 0.80</td>
</tr>
<tr>
<td>300</td>
<td>0.0, 0.05</td>
<td>0.86, 0.02</td>
<td>0.88, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.82, 0.1</td>
<td>0.85, 0.06</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.74, 0.21</td>
<td>0.82, 0.13</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.56, 0.80</td>
<td>0.74, 0.73</td>
</tr>
<tr>
<td>400</td>
<td>0.0, 0.05</td>
<td>0.85, 0.02</td>
<td>0.86, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.82, 0.09</td>
<td>0.85, 0.05</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.74, 0.20</td>
<td>0.81, 0.11</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.56, 0.78</td>
<td>0.73, 0.71</td>
</tr>
<tr>
<td>500</td>
<td>0.0, 0.05</td>
<td>0.83, 0.02</td>
<td>0.85, 0.005</td>
</tr>
<tr>
<td></td>
<td>0.0, 0.1</td>
<td>0.80, 0.08</td>
<td>0.84, 0.04</td>
</tr>
<tr>
<td></td>
<td>0.05, 0.1</td>
<td>0.73, 0.18</td>
<td>0.80, 0.1</td>
</tr>
<tr>
<td></td>
<td>0.1, 0.15</td>
<td>0.56, 0.76</td>
<td>0.72, 0.70</td>
</tr>
</tbody>
</table>

copy number state distribution, and the assumption of a proportion of affected cells.\(^5\)

These differences between the datasets likely account for the differences seen in the results, so it is important to know how these properties may affect results for microarray experiments.

Our implementation of the Triangulator algorithm required approximately

\(^5\)The simulation model used in our experiments does not assume that all cells in a culture are affected by copy number changes.
5m30s to process 500 simulated cell lines on a 3 GHz, 15 GB RAM server. Conversely, the Matlab implementation of CBS required approximately 6h40min on a 2.4 GHz, 3 GB machine. Although the Triangulator algorithm was implemented on a more powerful machine, the difference in computation time is still vast.

4.1.4 Discussion

The results suggest that the analysis of aCGH data can be performed very efficient and effectively on datasets where one can expect the signal-to-noise ratio to be high. In such cases, our method outperformed CBS. However, the CBS algorithm has better performance on noisier datasets. However, for datasets with low to moderate noise, the problem of detecting change points is still not trivial.

4.2 Detecting large structural variants at base pair level in NGS data

As stated beforehand, NGS technology is useful for detecting large structural variants at a higher resolution than the aCGH platform. It also allows for the detection of copy neutral events such as balanced translocations and inversions. To detect large structural variants, we use the combined general approach of 1) collecting groups of discordant read pairs, and then 2) searching for soft-clipped reads near these groups. We present algorithms to identify deletions, translocations, tandem repeat breakpoints, and inversions by using this framework. We then present a method to find double minute chromosomes that makes predictions by integrating copy-number predictions with NGS breakpoint predictions.
4.2.1 Methods

4.2.1.1 Requirements of a valid discordant read pair cluster

For each of the SV breakpoint detection algorithms, our programs take as input an alignment file in SAM format [54]. Discordant read pairs and soft-clipped reads are first extracted, and then the program looks for groups of overlapping discordant read pairs that adhere to certain constraints. Consider a structural variant whose first breakpoint is on chromosome $i$, and whose second breakpoint is on chromosome $j$. Chromosomes $i$ and $j$ could be the same chromosome, or they could be non-homologous. If such a structural variant exists, then there will ideally be a group of discordant read pairs that support the event. This “cluster” of read pairs should satisfy the following general criteria for all discordant pairs:

1. There must be a collection of reads $R(i)$ that map to chromosome $i$.

2. There must be a collection of reads $R(j)$ that map to chromosome $j$, where the reads of $R(j)$ are the mates of the reads in $R(i)$.

3. All reads in $R(i)$ must be mapped closely together and to the same strand.

4. All reads in $R(j)$ must be mapped closely together and to the same strand.

These criteria apply to all discordant read pair clusters, regardless of type. The criteria are explained further in the following subsections.
4.2.1.2 Interchromosomal variants

We have developed a method called “Bellerophon” to detect interchromosomal structural variants. To find such variants, the method searches for discordant read pairs that are chimeric, meaning that their mates map to different chromosomes. Interchromosomal variants include translocations and interchromosomal insertions. Regarding the criteria listed in Subsection 4.2.1.1, they are also applicable to finding interchromosomal breakpoints. Criteria 1 and 2 are straightforward because a true variant results in the fusion of two nonhomologous chromosomes. Mate pairs in the cluster must span the chimeric breakpoint, which results in a group of reads mapping to chromosome $i$, and their mates mapping to chromosome $j$. To understand criterion 3, consider the first chimeric pair that spans a particular breakpoint. Let’s call this pair $p$. When $p$ is mapped to the reference, there is no mapped distance information between its reads since they map to different chromosomes. After mapping, however, we expect that the distance between the first read of $p$ and the translocation breakpoint will be within $L$ base pairs, where $L = \text{mean} + k \times \text{stdev}$, and mean is the average separation distance between mapped read pairs in the dataset, stdev is the standard deviation of mapped distances, and $k$ is some constant, which for Bellerophon is a user defined value. Fig. 4.3 illustrates a chimeric cluster that implies a true translocation. Since the first encountered read in the cluster is within $L$ base pairs of the breakpoint, then subsequent chimeric read pairs must also map within $L$ base pairs of the breakpoint. It follows from this observation that all reads in $R(i)$ will map within $L$ base pairs of each other. This also holds the for the reads in $R(j)$. Furthermore, all the reads in a set must map to a common strand, because if the cluster implies a
true translocation, then one of the three scenarios must have occurred.

To predict likely translocations, Bellerophon must first find the clusters of chimeric read pairs. It does so by finding collections of chimeric reads that satisfy the criteria described in Subsection 4.2.1.1. Essentially, the algorithm collects all chimeric read pairs that map closely together on both ends. The pairs must also share the same two participating chromosomes. If such a collection of read pairs is found, then it possibly denotes a true chromosomal fusion. After a cluster is found by the algorithm, it must then determine which chromosomal arms create the fusion. It does so by examining the orientation of the aligned reads in a cluster, as shown in Fig. 4.4.

![Figure 4.3: Translocation captured by a cluster of three chimeric read pairs. The first set of reads map to the forward strand of chromosome $i$, and the second set map to the reverse strand of chromosome $j$. The distance between the outermost reads to the breakpoint are $D_1$ and $D_2$, for chromosomes $i$ and $j$ respectively. These distances must be less than or equal to $\text{mean} + k \times \text{stdev}$.](image-url)
1. p-arm to q-arm fusion
2. p-arm to p-arm fusion
3. q-arm to q-arm fusion

The mapping orientation of the read pairs depends on which chromosomal arms formed the translocation. Fig. 4.4 illustrates this. Lastly, there is criterion 4, but the requirements are the same as that of criterion 3.

![Diagram showing three types of translocations and mapping orientations](image)

Figure 4.4: The three types of translocations and the mapping orientations that result when a pair spans the breakpoint. These orientations assume that Illumina technologies were used in sequencing. Bellerophon deduces the type of fusion based on the mapping orientation of the pairs in a cluster.

After finding clusters that could imply a translocation, Bellerophon performs its breakpoint resolution step. In this step, the program will attempt to find the precise location on the genome where the chromosomal fusion occurred. This is an improvement over methods that only used paired-end reads, as such methods cannot accurately predict the true breakpoints.

To understand this step of the program, consider a cluster of chimeric pairs $K$ which was produced by the clustering algorithm referenced in the previous section,
and which has participating chromosomes $i$ and $j$. Because of criteria 3 and 4 of the clustering step, we expect that all chromosome $i$ reads of $K$ will map to within $L$ base pairs of the true chromosome $i$ breakpoint. Similarly, the chromosome $j$ reads will map to within $L$ base pairs of the true chromosome $j$ breakpoint.

Mapping the breakpoints works as follows: let $clip(x, y)$ be a soft clipped read where the aligned portion maps to chromosome $x$, and the clipped portion maps to chromosome $y$. Assuming the method is analyzing chromosome $i$ alignments, then once the clusters are formed, Bellerophon extends a window $W$ from the outermost read in $R(i)$ to the direction nearest the breakpoint. The size of this window is $L$, and the direction of the window extension depends on the type of chromosome fusion that the method is searching for. The method then performs the same step for chromosome $j$; it extends a window $X$ from the outermost read in $R(j)$ to the direction nearest the breakpoint. The size of $X$ is also $L$. Within windows $W$ and $X$, Bellerophon then searches for soft-clipped reads, because if there exists a true chimeric breakpoint $b$, then there should be reads at $b$ that partially align to both chromosomes $i$ and $j$. Specifically, we should have a collection of both $clip(i, j)$ and $clip(j, i)$ reads at $b$. Fig. 4.5 illustrates how soft-clipped reads are formed, and how they align to both sides of the variant boundary.

Now that the program has the clipped reads, it continues to the realignment step. The mapped location of the clipped portion of the read is unknown. In other words, once the breakpoint is encountered from both sides, we have $clip(i, x)$ and $clip(j, x)$ reads, where $x$ is an unknown chromosome. In this step, the algorithm determines the precise location of the clipped subreads so that it can determine $x$ and
Figure 4.5: The formation of soft-clipped reads. Soft-clipped reads span the translocation boundary between chromosomes $i$ and $j$. As a result, these reads may align partially to chromosome $i$ and partially to chromosome $j$.

the read's coordinates too. For each soft-clipped read within windows $W$ and $X$, if the size of the clipped portion is at least 20 base pairs, then this portion is aligned to the human reference genome using BLAT [55]. We required a threshold of 20 bp because BLAT is best suited to align sequences that are at least 20 base pairs in length.

In this step, we attempt to find the precise translocation breakpoints on both sides of the fusion. On the chromosome $i$ side of the boundary, the aligned portion of the clipped reads obviously map to chromosome $i$, but the clipped portion of the read should map to chromosome $j$. Similarly for the chromosome $j$ side of the boundary, the clipped subreads will map to chromosome $i$, whereas the aligned part of the read will map to chromosome $j$. As a result, we have the following four sets of subreads: $\text{Align}(i)$, $\text{Align}(j)$, $\text{Clipped}(i)$, $\text{Clipped}(j)$. The $\text{Align}(i)$ set contains the subreads whose alignment is to chromosome $i$. $\text{Align}(j)$ set is similarly defined for chromo-
some $j$. The $Clipped(i)$ and $Clipped(j)$ sets contain the clipped subreads that align to chromosomes $i$ and $j$, respectively.

To predict the breakpoint location for chromosome $i$, we calculate the mode of the aligned coordinates of reads in the set $S = Clipped(i) \cup Align(i)$. The subreads in $S$ may not necessarily align to the same location (due to small indels or mutations), so we assume that the true breakpoint will be the coordinate with the most subreads aligned to it. This step is also performed on the set $T = Clipped(j) \cup Align(j)$ to predict the precise breakpoint on the chromosome $j$ side of the variant. This breakpoint refinement step is shared by all of our algorithms to find basic structural variants (described in subsections 4.2.1.4, 4.2.1.5, and 4.2.1.6).

We use clipped alignment information from both sides of the breakpoint because it is possible that a true breakpoint will be without a sufficient number of clipped reads on either side of the boundary. By using clipped reads from each side, we increase the chances that it is accurately captured from both sides. To call a candidate variant as a predicted interchromosomal variant, there must exist at least one soft-clipped read that realigns to the window $W$, or at least one soft-clipped read that realigns to window $X$. Given the large size of the genome, the probability that even a single clipped sub-read remaps to the windows $X$ and $W$ is presumably small if no structural variation has occurred.

4.2.1.3 Chimeric breakpoint classification

It may be desirable to not only find chromosomal fusion boundaries, but to also determine the precise nature of the fusion; does it imply a balanced translo-
cation, unbalanced translocation, or interchromosomal insertion? Most methods for finding chimeric breakpoints do not attempt to answer this question; they are focused on finding the boundaries instead of resolving their exact nature (Fig. 4.6). In some experiments, a researcher may want to determine exactly how a chimeric fusion occurred. If it is caused by an interchromosomal insertion (Fig. 4.7), then a chromosome \( i \) donated a contiguous segment to a non-homologous chromosome \( j \), and two chimeric breakpoints are formed from this transfer of material. It is also possible that during the exchange, the orientation of the inserted segment might be inverted with respect to the centromere. This is known as an inverted insertion, whereas in the case of no inversion, it is a direct insertion [26]. Bellerophon accounts for both cases. The classification algorithm is provided in the Appendix Figs 1 and 2. Since interchromosomal insertions can also create fusion genes, it may be useful to predict their existence for some experiments.

4.2.1.4 Deletions

To detect deletions, we employ a similar framework as that of Bellerophon, except that we collect read pairs that are discordant only by their mapped distance (as shown for the deletion case in Fig. 2.19). Specifically, we collect read pairs whose mapped distance is greater than \( L \), where \( L = \text{mean} + k \times \text{stdev} \). These values are defined as they were in the previous subsection. The criteria for discordant read pair clusters are also similar, except that chromosomes \( i \) and \( j \) are the same for this case (since deletions are intrachromosomal events). The deletion finding algorithm, called “Pegasus”, looks for clusters of overlapping discordant pairs such that the left reads
Figure 4.6: The uncertainty of chimeric breakpoint prediction. Many SV methods only attempt to find chimeric boundaries as shown above, but they do not attempt to classify them. The above event could denote a balanced translocation between chromosomes 5 and 9, or it could denote an interchromosomal insertion from chromosome 9 to 5.

Reference chromosome i

Chromosome i

Chromosome j

Figure 4.7: An interchromosomal insertion and its resulting mapping signatures. Chromosome $i$ donated a segment to chromosome $j$. The read pair in chromosome $i$ spans the donor site, and when it is mapped back to the reference, its mapped distance will be much greater than its original distance in sequencing. To infer an interchromosomal insertion, Bellerophon not only searches for mirroring records, but it searches for these long, anomalously mapped pairs (AP). If two mirroring records have no corresponding long APs, then the records could imply a balanced translocation.
in each cluster are within $L$ base pairs of each other, and the right reads are also within $L$ base pairs of each other. This is a necessary requirement because if a group of discordant pairs imply a true deletion event, then their mapped distances should be similar. When a group of overlapping discordant pairs is found, the program then searches for soft-clipped reads that are presumably near the SV breakpoints. It then extracts the soft-clipped portion of at least one read and remaps it to the reference genome using BLAT. To be predicted as a deletion, a cluster of overlapping discordant pairs must satisfy two criteria: 1) there must be at least $c$ read pairs in the cluster, which for Pegasus is 3 (by default), and 2) at least one soft-clipped read from either side of the event must remap within the cluster region, which is the region from the outermost read in the cluster towards the variant breakpoint. Fig. 4.8 depicts a deletion event that is supported by two discordant read pairs and a soft-clipped read.

![Diagram](image)

Figure 4.8: An interchromosomal insertion and its resulting mapping signatures. Chromosome $i$ donated a segment to chromosome $j$. The read pair in chromosome $i$ spans the donor site, and when it is mapped back to the reference, its mapped distance will be much greater than its original distance in sequencing. To infer an interchromosomal insertion, Bellerophon not only searches for mirroring records, but it searches for these long, anomalously mapped pairs (AP). If two mirroring records have no corresponding long APs, then the records could imply a balanced translocation.
4.2.1.5 Inversions

We apply the framework to inversions by collecting discordant read pairs with forward-forward or reverse-reverse orientation (assuming the reads were sequenced with Illumina technologies). These kinds of discordant read pairs indicate a possible inversion, as shown with the example in Fig. 2.19. Because inversions are intrachromosomal events, the clustering and breakpoint refinement procedures are nearly identical to that of the deletions, except that during the clustering phase, the discordant pairs of interest are those where the aligned mates have the same orientation.

4.2.1.6 Tandem repeats

We can apply the framework to detecting tandem repeat breakpoints as well. As shown in Fig. 2.19, a tandem repeat will produce discordant read pairs that have reverse-forward orientation. This occurs because read pairs that flank the internal breakpoints on the donor will have this orientation when alignment back to the reference. As before, we first collect clusters of discordant read pairs of this type, and we subsequently refine the breakpoints by aligning soft-clipped reads. It should be noted that this approach is suitable for finding tandem repeat breakpoints at base pair level, but it is insufficient for measuring the number of repeats in the tandem. Such a task would require an analysis of read depth, which is proportional to the number of times the segment of interest was copied.
4.2.1.7 Double minute chromosomes

In addition to the previous algorithms for SV detection, we propose a method for algorithmically identifying double minute chromosomes using NGS data. Double minute chromosomes have two key properties. First, they are comprised of several distinct genomic segments. When these segments adjoin, their breakpoints exude one of the four kinds of paired end signals: 1) deletions, 2) inversions, 3) interchromosomal, 4) or tandem repeat. The second distinctive property of double minutes is their propensity to amplify. Every segment comprising the double minute is thus an amplicon, and the copy numbers of these amplicons should be similar to one another. We take advantage of these properties to develop a unified approach to detecting double minutes. As stated previously, a method to detect complex SVs must have the ability to find basic SV breakpoints, and then to subsequently “group” breakpoints together as a single complex variant prediction. Our approach incorporates copy number variant predictions to find amplicons (like that seen in Fig. 2.1), and it takes a set of SV predictions from NGS data to find amplicons that are “linked” together, as would be seen in a double minute chromosome. These graph-based algorithms are provided in Figs. 4.9 and 4.10. The first algorithm constructs an *amplicon graph* $G$, and an auxiliary amplicon graph $H$. The graph $G$ is undirected, and each vertex represents an amplicon that was discovered as per line 3. An edge connects two vertices (i.e. amplicons) in this graph if for an NGS-based SV prediction, its predicted breakpoints are proximal to the predicted breakpoints for each amplicon$^6$. Fig. 4.11 shows how

---

$^6$We define “proximal” as being within $L = mean + k \times stdev$, where $mean$ and $stdev$ are the mean read pair mapping distance and its standard deviation, and $k$ is a user-defined constant.

55
an amplicon graph is created from amplicon predictions and SV predictions.

**Algorithm 1** Algorithm to construct amplicon graphs

1: \( G \leftarrow \emptyset \) \{undirected graph\}
2: \( H \leftarrow \emptyset \) \{directed graph\}
3: Extract set of amplicons \( A \) from copy number predictions
4: \textbf{for all} \( a \in A \) \textbf{do}
5: \hspace{1em} Add vertex \( v(a) \) to \( G \)
6: \hspace{1em} Add vertex \( v(a) \) to \( H \)
7: \textbf{end for}
8: Extract set of SV breakpoint predictions from NGS data
9: \textbf{for all} \( v, w \in V(G) \) such that \( v \neq w \) \textbf{do}
10: \hspace{1em} if a breakpoint prediction joins amplicons \( v \) and \( w \) \textbf{then}
11: \hspace{2em} Add edge \( (v, w) \) to \( G \)
12: \hspace{1em} \textbf{end if}
13: \textbf{end for}
14: Perform modified depth-first-search on \( G \) starting from arbitrary vertex
15: During the depth-first search:
16: \textbf{if} the current edge \( (u, v) \) is unexplored \textbf{then}
17: \hspace{1em} add directed edge \( (u, v) \) to \( H \) with weight 1
18: \textbf{end if}

Figure 4.9: Algorithm to build the amplicon graph \( G \) and auxiliary graph \( H \).
Algorithm 2 Algorithm to find all double minutes in the amplicon graph

Adjacent(u): return the set of vertices that are adjacent to vertex u

all_paths: for all vertices u, the set of all shortest paths from all vertices in Adjacent(u) to u

shortest_path(G, u, v): for a directed graph G, the shortest path from a vertex u to a vertex v

Let SCC be the set of all strongly connected components in H

5: for all c ∈ SCC do
   Run all-pairs-shortest-path algorithm on subgraph c
   all_paths ← ∅
   for all u ∈ c do
      for all v ∈ Adjacent(u) do
         Add shortest_path(c, v, u) to all_paths
      end for
   end for
   DM ← p, such that p is the path of minimum weight in all_paths
   max ← {mapping coverage of amplicon with greatest coverage in DM}
15: min ← {mapping coverage of amplicon with least coverage DM}
   if |max − min| < average mapping coverage then
      Predict DM as a DOUBLE MINUTE
      Remove from H all vertices in DM
   end if
20: end for

Figure 4.10: Algorithm to find all double minutes that are represented in the amplicon graph.

Figure 4.11: Conversion of NGS breakpoints and amplicons to an amplicon graph. The segments in the top image are amplicons with elevated copy number. The colored lines represent SV predictions for a translocation (green line), and a deletion (blue line). The corresponding amplicon graph is depicted in the bottom image. Vertices are amplicons and edges are NGS-based SV breakpoint predictions.

Double minute chromosomes consist of consecutively-joined segments that are
highly amplified. Our algorithms exploit the fact that double minutes are circular, and thus in the amplicon graph, the corresponding subgraph representing a true double minute should contain a simple path from some vertex back to itself. Lines 1-13 in Algorithm 1 are straightforward. It simply builds the amplicon graph $G$, and it adds the same vertices in $G$ to the auxiliary graph $H$. In line 13, we perform a modified depth-first search (DFS) on the graph $G$ by keeping track of whether a currently explored edge has already been traversed. If not, then we add the corresponding directed edge to $H$ with an edge weight of 1. This DFS is performed because we ultimately want our algorithm to identify cyclical subgraphs. If there is some simple path from a vertex $v$ back to itself (excluding the trivial case where the path only contains $v$), a DFS will encounter a series of edges that represent a path from that vertex back to itself. While searching $G$, we add a weighted directed edge to $H$ if the current edge in the search is unexplored. This ensures that for the graph $G$, every vertex reachable from itself will have a corresponding directed path in $H$ with a path to itself. If such a path can be identified, then it could indicate a possible double minute.

In Algorithm 2, the method processes the auxiliary graph $H$ and determines whether or not it contains double minute chromosomes. It does this by first collecting all of the strongly connected components (SCC) in $H$, because a double minute is an SCC, or should at least be captured in a SCC. In line 5 of Algorithm 2, the method iterates through each SCC to see if it contains a double minute. It first runs an all-pairs-shortest-path algorithm on each retrieved SCC, and for each vertex $u$ in

\footnote{It is not necessarily true that all SCCs in $H$ are double minutes}
an SCC, it searches for a shortest path from \( v \) to \( u \), where \( v \) is a vertex adjacent to \( u \). This step is performed because a double minute should contain a simple cycle from any vertex back to itself. However, we do not query the shortest path matrix for a path from \( u \) back to itself, because the shortest path from a node to itself is simply that node. To find a double minute, the method finds the shortest path cycle that minimizes the total edge weight. Thus, we assume that a double minute will be a cycle containing the fewest vertices (i.e. amplicons). This cycle is stored to \( DM \) in line 13. As previously stated, double minutes must consist of amplicons that have similar copy number. In NGS experiments, the copy number of a region in a donor genome is proportional to the sequence coverage in that region. Thus, each amplicon should also have similar mapping coverage. We check this condition by assessing the difference between the amplicon with the highest mapping coverage, and the amplicon with the lowest mapping coverage. This is performed in lines 15-17. If this difference is less than the average mapping coverage, then we predict the entire cycle in \( DM \) as a double minute.

4.2.2 Experimental design
4.2.2.1 Interchromosomal variants: two simulated datasets

In our first experiment, we applied the Bellerophon program to simulated data to detect interchromosomal events. We used two simulated datasets to test our program’s ability to 1) detect translocation breakpoints and to 2) accurately predict the location of the translocation breakpoints. For the first dataset, we created a simulated test genome by inserting into the human reference genome the variants listed in
Table 4.6. Since balanced translocations are not always entirely reciprocal, we added a 1000 bp duplication to the p-arm of chromosome 6. Duplications and deletions at reciprocal translocation breakpoints occur in some cancers [56], so Bellerophon allows for reciprocal translocation breakpoints to overlap by at most 1 Mb, or to be separated by at most 1 Mb (by default). For the second dataset, we inserted into the reference genome the variants listed in Table 4.7.

Table 4.6: Structural variants inserted into the first simulated dataset. U = unbalanced translocation, II = interchromosomal insertion, and B = balanced translocation. For the “II” and “B” variants, the partner breakpoints are listed consecutively. For “II” variants, the donor chromosome and its breakpoint are bolded. Note that the chr3 and chr6 balanced translocation contains a 1000-bp duplication, so it is not entirely reciprocal.

<table>
<thead>
<tr>
<th>Chr1</th>
<th>Bkpt1</th>
<th>Strand1</th>
<th>Chr2</th>
<th>Bkpt2</th>
<th>Strand2</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>73,000,000</td>
<td>+</td>
<td>11</td>
<td>63,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>5</td>
<td>40,000,000</td>
<td>+</td>
<td>2</td>
<td>140,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>7</td>
<td>11,000,000</td>
<td>+</td>
<td>12</td>
<td>45,000,000</td>
<td>-</td>
<td>U</td>
</tr>
<tr>
<td>10</td>
<td>5,000,000</td>
<td>-</td>
<td>20</td>
<td>15,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>16</td>
<td>6,000,000</td>
<td>-</td>
<td>18</td>
<td>12,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>4</td>
<td>9,000,000</td>
<td>+</td>
<td>17</td>
<td>17,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>3</td>
<td>35,000,000</td>
<td>+</td>
<td>6</td>
<td>14,000,000</td>
<td>+</td>
<td>B</td>
</tr>
<tr>
<td>6</td>
<td>14,001,000</td>
<td>+</td>
<td>3</td>
<td>35,000,001</td>
<td>+</td>
<td>B</td>
</tr>
<tr>
<td>13</td>
<td>45,000,000</td>
<td>+</td>
<td>14</td>
<td>30,000,000</td>
<td>+</td>
<td>II</td>
</tr>
<tr>
<td>14</td>
<td>30,200,000</td>
<td>+</td>
<td>13</td>
<td>45,000,001</td>
<td>+</td>
<td>II</td>
</tr>
<tr>
<td>1</td>
<td>105,000,000</td>
<td>-</td>
<td>22</td>
<td>25,000,000</td>
<td>+</td>
<td>II</td>
</tr>
<tr>
<td>22</td>
<td>25,000,001</td>
<td>+</td>
<td>1</td>
<td>105,600,000</td>
<td>-</td>
<td>II</td>
</tr>
</tbody>
</table>

For the first dataset, we created simulated paired-end sequence reads from this synthetic data using wgsim [57]. We generated the dataset with 40X sequence read coverage and 100 base pair (bp) reads. It also had a 400 bp average insert size with a standard deviation of 80. The mutation rate was set at 0.001, and among the
<table>
<thead>
<tr>
<th>Chr1</th>
<th>Bkpt1</th>
<th>Strand1</th>
<th>Chr2</th>
<th>Bkpt2</th>
<th>Strand2</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>41,000,000</td>
<td>+</td>
<td>18</td>
<td>50,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>13</td>
<td>31,000,000</td>
<td>+</td>
<td>20</td>
<td>43,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>9</td>
<td>21,000,000</td>
<td>-</td>
<td>17</td>
<td>60,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>21</td>
<td>30,000,000</td>
<td>+</td>
<td>2</td>
<td>35,000,000</td>
<td>-</td>
<td>U</td>
</tr>
<tr>
<td>11</td>
<td>11,000,000</td>
<td>+</td>
<td>12</td>
<td>67,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>16</td>
<td>23,000,000</td>
<td>+</td>
<td>7</td>
<td>44,000,001</td>
<td>+</td>
<td>B</td>
</tr>
<tr>
<td>7</td>
<td>44,000,000</td>
<td>+</td>
<td>16</td>
<td>23,000,001</td>
<td>+</td>
<td>B</td>
</tr>
<tr>
<td>6</td>
<td>92,000,000</td>
<td>-</td>
<td>10</td>
<td>65,000,000</td>
<td>+</td>
<td>U</td>
</tr>
<tr>
<td>19</td>
<td>35,000,000</td>
<td>+</td>
<td>14</td>
<td>55,000,000</td>
<td>-</td>
<td>U</td>
</tr>
</tbody>
</table>

Table 4.7: Structural variants inserted into the second simulated dataset.

After creating the sequence reads, we aligned them to the human reference genome (NCBI 36) using BWA. After acquiring the alignment results, we created four more datasets by randomly “down sampling” the original alignments at the following rates: 75%, 50%, 25%, and 10%. Each rate is the probability that an aligned mate pair (or single anchoring read) from the original SAM file would be included in the new sampled SAM file. Thus, the resulting files had average coverage of approximately 30X, 20X, 10X, and 4X. After this step, the five datasets were analyzed with the following programs: Bellerophon, GASV, Breakdancer, SVDetect, and CREST. On these datasets, we compared each program’s ability to predict breakpoints by measuring sensitivity, specificity, and average breakpoint error. For a given SV prediction, the breakpoint error is defined as the difference in base pairs between the true variant boundary and the predicted variant boundary. After identifying the in-

---

8We did not use GASVPro since it is not equipped to handle translocations
individual breakpoints, we applied the classification algorithm to the prediction results from the 40X coverage alignments.

### 4.2.2.2 Interchromosomal variants: primary prostate cancer datasets

In our second experiment, we investigated Bellerophon’s ability to predict translocations in two prostate cancer datasets PR-0508 and PR-1783 [31]. These datasets were sequenced using the Illumina GA II sequencer at 30X haploid coverage. The insert sizes were approximately 400 bp and the read lengths were 101 bp. After removing replicate artifacts using the Picard suite, we aligned the sequence reads to the human reference genome. We then applied the same experimental design to the cancer data that was applied to the simulated data described in the previous section. Thus, the resulting sampled alignment files had read depth of approximately 22.5X, 15X, 7.5X, and 3X for the 75%, 50%, 25%, and 10% sampling rates respectively. The list of interchromosomal breakpoints for each dataset is provided in the original study [31].

### 4.2.2.3 Deletions

To test our deletion breakpoint method, we performed two experiments in which we compared it to the CREST and BreakDancerMax programs. In the first experiment, we assessed the ability of each method to detect deletions of varying lengths in a simulated dataset. The breakdown of deletions by size is provided in Table 4.8. To generate the simulated data, we inserted the deletions into chromosome 9 of the human reference genome (NCBI build 36). Using Wgsim, we then generated
synthetic sequence reads at 40X coverage with 400bp mean inserts and standard deviation of 50. The read lengths were 100bp. The sequence reads were then aligned to chromosome 9 of the reference genome using BWA. After performing alignment, we created another simulated dataset by downsampling the alignment results; we randomly selected read pairs and singleton reads at a rate of 25% for inclusion in the new dataset. The goal of this downsampling was to assess the performance of each method in the presence of low sequence coverage. Since the method uses both discordant reads and single reads in predicting variants, we hypothesized that it would be more resilient than the other methods to varying levels of read depth. After creating the downsampled dataset, we measured the sensitivity, specificity, F-score, and average breakpoint error for each program.

Table 4.8: Summary of deletions inserted into chromosome 9. There were a total of 102 simulated deletions.

<table>
<thead>
<tr>
<th>Chr1 Length</th>
<th>Bkpt1</th>
<th>Strand1</th>
<th>Chr2</th>
<th>Bkpt2</th>
<th>Strand2</th>
<th>Type</th>
<th># deletions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1kb</td>
<td>10kb</td>
<td>25kb</td>
<td>50kb</td>
<td>100kb</td>
<td>100kb</td>
<td>Total</td>
<td>102</td>
</tr>
</tbody>
</table>

In our second experiment we assessed the performance of each method on the two aforementioned prostate cancer datasets PR-0508 and PR-1783. We aligned the sequence reads to the human reference genome and applied the same experimental setup used for the chromosome 9 simulated data, only this time we filtered out replicates using the Picard Suite\textsuperscript{9}. With these alignment results, we performed the same experimental setup described for the simulated data.

\textsuperscript{9}The real data contained many more replicates than the simulated data
4.2.2.4 Inversions

We aligned the PR-0508 and PR-1783 datasets to the reference genome and we applied our inversion breakpoint algorithm on these data to see how well it captured the inversion breakpoints that were published in the study by Berger et al. For these data, there were 13 inversion breakpoints reported for PR-1783, and 2 breakpoints reported for the PR-0508 sample. We ran our algorithm on each dataset and measured the number of true breakpoints captured as well as the location of the predicted breakpoints.

4.2.2.5 Tandem repeats

We applied the tandem repeat breakpoint algorithm to the PR-0508 and PR-1783 datasets. For the PR-0508 dataset, Berger et al. reported 3 tandem repeat breakpoints, while for the PR-1783 dataset, they reported 12 breakpoints. As with the inversion algorithm, we measured the number of true breakpoints captured for each dataset and the location of the predicted breakpoints.

4.2.2.6 Double minute chromosomes

We tested our double minute prediction algorithm on a simulated NGS dataset that was constructed based on a chromosome 2 double minute presented by Rausch et al. [33]\textsuperscript{10}. This double minute consists of 10 amplicons connected by deletion, tandem repeat, and inversion breakpoints. For this complex event, we took the coordinates of the amplicons reported by Rausch and we extracted these regions from

\textsuperscript{10}Due to time constraints, we did not use the real data from Rausch’s study.
the human reference genome. We then appended together consecutive amplicons in
the double minute, per the description provided in the original study. This appending
created the SV breakpoints that connected each amplicon. After performing this
step, we “amplified” the double minute 10-fold; we made 10 copies of each amplicon
and stored them to a FASTA file. We then added the reference chromosome 2 (build
36) to this FASTA file and generated sequence reads on this synthetic sequence using
wgsim. The reads were “sequenced” at 30X coverage with 100 bp reads and 400 bp
fragment lengths, with a standard deviation of 80. As with the previous experiments,
the mutation rate was set to 0.001 and the indel rate was set to 0.15.

We aligned the aforementioned sequence reads to chromosome 2 of the hu-
man reference using Bowtie2 [58] \(^{11}\). To detect amplicons, we provided the result-
ning SAM file to RDXplorer, which is a program to detect copy number variants in
NGS data [59]. This step is necessary to fulfill the line 3 statement in Algorithm
1 (Fig. 4.9)\(^{12}\). After observing the list of predicted amplicons, we found that many
predicted amplicons were really from a single amplicon, but with discontinuities in
the copy number prediction. This could be caused by small fluctuations in mapping
coverage, which is a common occurrence, regardless of the copy number state of a
region. However, it is also possible that a discontinuity in coverage is due to a le-
gitimate structural variant, like a deletion. Thus, we merged consecutive amplicons

\(^{11}\)We did not use BWA for this experiment because the alignment results were inexplicably erro-
neous. It could be due to a bug in the program. Furthermore, we did not use Bowtie because it
does not produce soft-clipped reads.

\(^{12}\)We did not use CGH-Triangulator to detect amplicons because it cannot predict the copy number
state of a region (i.e. "gain", "loss", "neutral"). However, any algorithm that predicts copy number
state can be used for the step in line 3 of Algorithm 1
that were 1) within 10,000 bp of each other, 2) had predicted copy numbers that were within 2 copies of each other, per RDXplorer, and 3) if the merge boundary was not proximal to an SV breakpoint prediction. After acquiring the amplicons and applying this simple merging procedure, we analyzed the SAM file with our deletion, tandem repeat, and inversion breakpoint detection algorithms. We did not run this dataset on our translocation detecting program since the simulated double minute contained only chromosome 2 amplicons. After this step, we applied our algorithms on this dataset to see if the method could identify the double minute.

4.2.3 Results
4.2.3.1 Interchromosomal variants: results on simulated datasets

The results of each method on the simulated datasets are provided in Tables 4.9 and 4.10. Visualized results are provided in Figs. 4.12 and 4.13. For the simulated data, all of the methods performed well with regards to specificity. However, Bellerophon had the highest total sensitivity across all datasets. As expected, CREST did not perform well on the 10% dataset. For this low-coverage dataset, the variant breakpoints were spanned by few individual reads. Because CREST only relies on the presence of soft clipped reads, it is susceptible to losses in sensitivity in such data, especially since it requires several soft clipped reads to trigger the assembly portion of its algorithm. Because Bellerophon uses paired reads in addition to soft-clipped reads, it does not require that many individual reads span a variant boundary. For Bellerophon, we required at least one soft-clipped read from either side of the breakpoint and at most five soft-clipped reads from both sides of the
breakpoint. This second requirement is for efficiency.

The paired-end methods performed well on the lower coverage datasets. However, as expected, they are unable to accurately call precise breakpoints. Tables 4.9 and 4.10 highlight the advantage of using a method like Bellerophon that uses both paired end and split read strategies for SV detection. Bellerophon’s breakpoint estimation performance was similar to that of CREST, but it was more resilient to decreasing levels of read depth. This is likely due to CREST’s dependence on several sequence reads containing the variant breakpoint, whereas Bellerophon only requires at least one such read. Because the cluster region is very small compared to the size of the genome, it is unlikely that even a single soft-clipped subread will remap to this region by chance.

<table>
<thead>
<tr>
<th>Method</th>
<th>40X</th>
<th>30X</th>
<th>20X</th>
<th>10X</th>
<th>4X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE</td>
<td>SP</td>
<td>ABE</td>
<td>SE</td>
<td>SP</td>
</tr>
<tr>
<td>B-phon</td>
<td>12/12</td>
<td>12/12</td>
<td>0.96</td>
<td>12/12</td>
<td>12/12</td>
</tr>
<tr>
<td>SVD</td>
<td>11/12</td>
<td>11/12</td>
<td>2.7-417</td>
<td>11/12</td>
<td>11/11</td>
</tr>
<tr>
<td>BD</td>
<td>12/12</td>
<td>12/12</td>
<td>195.3</td>
<td>12/12</td>
<td>12/12</td>
</tr>
<tr>
<td>GASV</td>
<td>12/12</td>
<td>12/12</td>
<td>105-226</td>
<td>12/12</td>
<td>12/12</td>
</tr>
<tr>
<td>CREST</td>
<td>12/12</td>
<td>12/12</td>
<td>0.9</td>
<td>11/12</td>
<td>11/11</td>
</tr>
</tbody>
</table>

Table 4.9: Simulated dataset 1 results (100 bp reads)

<table>
<thead>
<tr>
<th>Method</th>
<th>40X</th>
<th>30X</th>
<th>20X</th>
<th>10X</th>
<th>4X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE</td>
<td>SP</td>
<td>ABE</td>
<td>SE</td>
<td>SP</td>
</tr>
<tr>
<td>B-phon</td>
<td>9/9</td>
<td>9/9</td>
<td>3.5</td>
<td>9/9</td>
<td>9/9</td>
</tr>
<tr>
<td>BD</td>
<td>9/9</td>
<td>14/14</td>
<td>149.7</td>
<td>9/9</td>
<td>14/14</td>
</tr>
<tr>
<td>GASV</td>
<td>9/9</td>
<td>9/9</td>
<td>75-177</td>
<td>9/9</td>
<td>9/9</td>
</tr>
<tr>
<td>CREST</td>
<td>9/9</td>
<td>15/15</td>
<td>1.2</td>
<td>9/9</td>
<td>14/14</td>
</tr>
</tbody>
</table>

Table 4.10: Simulated dataset 2 results (75 bp reads)

Regarding classification results, the Bellerophon program correctly classified all of the interchromosomal events created in our simulated data. For the events
Figure 4.12: Visual results for the 100 bp simulated dataset. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions.

Figure 4.13: Visual results for the 75 bp simulated dataset. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions.
with partner predictions (i.e. “B” and “II” events), the partner breakpoints were correctly identified in all cases. As stated beforehand, Bellerophon can predict the presence of balanced translocations and complex interchromosomal insertions. GASV and SVDetect can predict balanced translocations, but for the 100 bp simulated data, they did not identify the chr3/chr6 reciprocal exchange. This was possibly due to the 1000 bp duplication at the chr6 breakpoint. However, GASV did correctly identify the reciprocal translocation in the 75 bp dataset.

4.2.3.2 Interchromosomal variants: results on prostate cancer datasets

The results on the prostate cancer datasets are provided in Tables 4.11 and 4.12. Visualized results are provided in Figs. 4.14 and 4.15. For the PR-0508 dataset, Bellerophon’s maximum sensitivity is 6/8, but one of the true events presented no signal for SV detection, so it was undetected by all five methods. Another event was captured by discordant read pairs, but its breakpoints were not spanned by soft-clipped reads. Thus, Bellerophon and CREST did not capture this event either. Overall, Bellerophon’s breakpoint accuracy was slightly less than CREST, but its sensitivity was greater in all cases. Many false positives were predicted by all the methods, which were largely due to repeating elements and sequences that were homologous to true cluster regions. From our own results, we noticed that many of our false positives involved breakpoint regions that were found in the centromere or in repeat-rich regions. They were also caused by highly polymorphic regions among non-homologous chromosomes, leading to the formation of “false” chimeric clusters. Despite this, the CREST program reported far fewer false positives than the other methods because
unlike the other methods, CREST exclusively uses soft-clipped reads to make its predictions. However, for breakpoints spanned by few soft-clipped reads, CREST will not perform well. In contrast, Bellerophon is less conservative in its calls since it requires fewer such reads. Thus, its false positive predictions are higher. However, by combining the paired-read and single-read signatures, Bellerophon achieved higher sensitivity than CREST, but a lower false positive rate compared to the paired-read methods. Essentially, the combined strategy maximizes sensitivity while mitigating the loss of specificity.

After running the classification step on the predictions from the prostate cancer datasets, we found 40 and 14 interchromosomal insertions in the PR-1783 and the PR-0508 datasets, respectively. However, these events involved at least one breakpoint that was located in the centromeres of their respective chromosomes, so we did not regard them as high confidence predictions. The PR-0508 dataset contained two apparent balanced translocations, and both were correctly identified by our method. The PR-1783 dataset contained one apparent balanced translocation that was also correctly identified. For the two datasets used in our experiments, the original studies did not state exactly how the interchromosomal breakpoints were formed, so we did not have “ground truth” information on which to compare our classification results. This is less problematic for balanced translocations, because their existence can be inferred by observing the breakpoint data. Bellerophon is better suited to locate large interchromosomal insertions (as seen with the simulated data). Regarding insertions, the lack of high quality predictions in the real data could indicate that many true events could be smaller in size, and thus the program would be less sensitive to their
presence.

As stated beforehand, the combined strategy of Bellerophon is advantageous because it combines the strengths of paired-read and split read methods. Moreover, the weaknesses of both approaches are mitigated by the simultaneous use of both signals. In particular, for predicting precise breakpoints, Bellerophon only needs one soft-clipped read on each side of a fusion boundary. The CREST method performs best when many such reads are present. Compared to the other paired-end methods, Bellerophon had better specificity and near equal sensitivity, while having superior breakpoint accuracy. Given this, Bellerophon would perform well in both low and high coverage datasets. For the cancer datasets, some of the interchromosomal breakpoints identified by our method were involved in gene fusions as specified in the original study. Thus, the method is a useful tool for discovering and characterizing gene fusions caused by interchromosomal structural variants.

Table 4.11: Results on the PR-0508 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>30X SE</th>
<th>30X SP</th>
<th>30X ABE</th>
<th>22.5X SE</th>
<th>22.5X SP</th>
<th>22.5X ABE</th>
<th>15X SE</th>
<th>15X SP</th>
<th>15X ABE</th>
<th>7.5X SE</th>
<th>7.5X SP</th>
<th>7.5X ABE</th>
<th>3X SE</th>
<th>3X SP</th>
<th>3X ABE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>7/8</td>
<td>7/738</td>
<td>7.6-293</td>
<td>7/8</td>
<td>7/544</td>
<td>12.4-282</td>
<td>7/8</td>
<td>7/367</td>
<td>27-265</td>
<td>4/8</td>
<td>4/169</td>
<td>35.3-247</td>
<td>2/8</td>
<td>2/49</td>
<td>30.3-278</td>
</tr>
<tr>
<td>BD</td>
<td>7/8</td>
<td>9/490</td>
<td>141.2</td>
<td>7/8</td>
<td>8/343</td>
<td>165.1</td>
<td>5/8</td>
<td>5/233</td>
<td>131</td>
<td>2/8</td>
<td>2/112</td>
<td>161</td>
<td>0/8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CREST</td>
<td>5/9</td>
<td>5/55</td>
<td>1.1</td>
<td>5/9</td>
<td>5/43</td>
<td>1.1</td>
<td>4/9</td>
<td>4/37</td>
<td>1.0</td>
<td>2/8</td>
<td>2/17</td>
<td>0.5</td>
<td>0/9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 4.12: Results on the PR-1783 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>30X SE</th>
<th>30X SP</th>
<th>30X ABE</th>
<th>22.5X SE</th>
<th>22.5X SP</th>
<th>22.5X ABE</th>
<th>15X SE</th>
<th>15X SP</th>
<th>15X ABE</th>
<th>7.5X SE</th>
<th>7.5X SP</th>
<th>7.5X ABE</th>
<th>3X SE</th>
<th>3X SP</th>
<th>3X ABE</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-phon</td>
<td>9/9</td>
<td>9/290</td>
<td>7.7</td>
<td>9/9</td>
<td>9/212</td>
<td>8.3</td>
<td>7/9</td>
<td>7/156</td>
<td>2.6</td>
<td>5/9</td>
<td>5/86</td>
<td>4.7</td>
<td>1/9</td>
<td>1/39</td>
<td>0</td>
</tr>
<tr>
<td>BD</td>
<td>9/9</td>
<td>12/408</td>
<td>171.0</td>
<td>8/9</td>
<td>8/305</td>
<td>180</td>
<td>6/9</td>
<td>6/210</td>
<td>167</td>
<td>2/9</td>
<td>2/114</td>
<td>137</td>
<td>0/9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CREST</td>
<td>5/9</td>
<td>5/60</td>
<td>3.2</td>
<td>5/9</td>
<td>5/49</td>
<td>3.2</td>
<td>3/9</td>
<td>3/38</td>
<td>2.2</td>
<td>1/9</td>
<td>1/15</td>
<td>1.5</td>
<td>0/9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
4.2.3.3 Deletions: results on chromosome 9 simulated data

The results for each method are provided in Tables 4.13 and 4.14. As shown, Pegasus had a higher F-score than the other methods on both the original and 25% datasets. It also had the highest specificity of the methods. Although BreakDancer-Max had lower breakpoint error than Pegasus on the full dataset, it had higher error on the 25% dataset.

Table 4.13: Sensitivity and specificity on the simulated dataset. BD = BreakDancer, PEG = Pegasus, CRST = CREST. Some methods have redundant predictions, but these were not counted against their specificity.

<table>
<thead>
<tr>
<th>Chr1</th>
<th>Bkpt1</th>
<th>Strand1</th>
<th>Chr2</th>
<th>Bkpt2</th>
<th>Strand2</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>PEG SE</td>
<td>PEG SP</td>
<td>CRST SE</td>
<td>CRST SP</td>
<td>BD SE</td>
<td>BD SP</td>
</tr>
<tr>
<td>Original</td>
<td>101/102</td>
<td>106/106</td>
<td>97/102</td>
<td>216/226</td>
<td>101/102</td>
<td>101/141</td>
</tr>
<tr>
<td>25%</td>
<td>99/102</td>
<td>99/99</td>
<td>76/102</td>
<td>96/99</td>
<td>100/102</td>
<td>100/109</td>
</tr>
</tbody>
</table>
Figure 4.15: Visual results for PR-1783. The size of the circles is inversely proportional to the F1 score of the breakpoint predictions.

Table 4.14: F-scores and average breakpoint error for each method on the simulated datasets.

<table>
<thead>
<tr>
<th>Chr1</th>
<th>Bkpt1</th>
<th>Strand1</th>
<th>Chr2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Pegasus</td>
<td>BreakDancerMax</td>
<td>CREST</td>
</tr>
<tr>
<td>Original</td>
<td>0.995, 8.5</td>
<td>0.831, 6.4</td>
<td>0.953, 1.7</td>
</tr>
<tr>
<td>25%</td>
<td>0.985, 2.7</td>
<td>0.948, 20.6</td>
<td>0.843, 1.4</td>
</tr>
</tbody>
</table>

4.2.3.4 Deletions: results on prostate cancer datasets

The result of the experiment for the cancer datasets is provided in Tables 4.15-4.18. As shown, all three methods predicted many more false positives on the cancer dataset in contrast to the simulated dataset. These false positives can occur due to chance, but they are mostly the consequence of repeating elements in the genome and intrachromosomal sequence similarity. Despite this, Pegasus achieved the highest
F-score of all the methods for both samples on the 25% datasets. CREST had the lowest number of false positives, but it also had a sharp dropoff in sensitivity for the 25% datasets. BreakDancerMax had good performance on sensitivity, but it called many more false positives than the other two methods. Just like the results for the simulated data, BreakDancerMax expectedly had the worse breakpoint prediction accuracy of the methods.

Table 4.15: Sensitivity and specificity results on PR-0508.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PEG SE</th>
<th>PEG SP</th>
<th>CRST SE</th>
<th>CRST SP</th>
<th>BD SE</th>
<th>BD SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>19/22</td>
<td>19/22232</td>
<td>19/22</td>
<td>19/2035</td>
<td>22/22</td>
<td>22/3974</td>
</tr>
</tbody>
</table>

Table 4.16: F-score and breakpoint error on PR-0508.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PEG SE</th>
<th>PEG SP</th>
<th>CRST SE</th>
<th>CRST SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.017, 1.1</td>
<td>0.011, 22.5</td>
<td>0.018, 0.8</td>
<td></td>
</tr>
<tr>
<td>25%</td>
<td>0.013, 1.0</td>
<td>0.012, 33.9</td>
<td>0.008, 0.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.17: Sensitivity and specificity results on PR-1783.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PEG SE</th>
<th>PEG SP</th>
<th>CRST SE</th>
<th>CRST SP</th>
<th>BD SE</th>
<th>BD SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>34/39</td>
<td>34/2819</td>
<td>23/39</td>
<td>25/1886</td>
<td>36/39</td>
<td>36/3281</td>
</tr>
<tr>
<td>25%</td>
<td>9/39</td>
<td>9/1202</td>
<td>2/39</td>
<td>2/494</td>
<td>14/39</td>
<td>14/2777</td>
</tr>
</tbody>
</table>

Table 4.18: F-score and breakpoint error on PR-0508.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PEG SE</th>
<th>PEG SP</th>
<th>CRST SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.024, 8.3</td>
<td>0.022, 26.6</td>
<td>0.026, 1.4</td>
</tr>
<tr>
<td>25%</td>
<td>0.015, 2.16</td>
<td>0.01, 54.6</td>
<td>0.008, 2.25</td>
</tr>
</tbody>
</table>
4.2.3.5 Inversions: results on prostate cancer datasets

The results of the inversion breakpoint algorithm on the prostate cancer datasets are provided in Tables 4.19 and 4.20. This table also lists the variants reported by Berger. Many of our results were filtered out because there were not enough soft-clipped reads (we required at least one from each side of the breakpoints). However, our reported results were highly concordant with those reported in the original study, indicating that the algorithm is suitable for detecting inversion breakpoints at high resolution.

Table 4.19: PR-0508 results. Inversion breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Berger et al.</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr</td>
<td>Pos 1</td>
<td>Pos 2</td>
</tr>
<tr>
<td>7</td>
<td>78,522,979</td>
<td>78,969,793</td>
</tr>
<tr>
<td>7</td>
<td>78,504,428</td>
<td>78,959,636</td>
</tr>
</tbody>
</table>

4.2.3.6 Tandem repeat breakpoints: results on prostate cancer datasets

For the tandem repeat breakpoints, the comparison results of our method on the prostate cancer datasets are provided in Tables 4.21 and 4.22. It should be noted that this is only an assessment of the ability of our program to detect tandem repeat breakpoints (using the R-F discordant read pair signature). Other types of structural variants, like transposons, can also produce this signature, so to truly detect tandem repeats, some read depth analysis may be required.
Table 4.20: PR-1783 results. Inversion breakpoints reported by Berger et al., and the breakpoints predicted by our inversion algorithm.

<table>
<thead>
<tr>
<th>Chr</th>
<th>Berger et al.</th>
<th>Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chr</td>
<td>Pos 1</td>
</tr>
<tr>
<td>1</td>
<td>71,889,465</td>
<td>72,791,865</td>
</tr>
<tr>
<td>3</td>
<td>84,760,066</td>
<td>85,313,569</td>
</tr>
<tr>
<td>3</td>
<td>85,699,550</td>
<td>86,247,542</td>
</tr>
<tr>
<td>5</td>
<td>96,381,150</td>
<td>96,482,482</td>
</tr>
<tr>
<td>6</td>
<td>124,878,148</td>
<td>124,975,105</td>
</tr>
<tr>
<td>6</td>
<td>124,890,582</td>
<td>124,984,001</td>
</tr>
<tr>
<td>7</td>
<td>7,865,326</td>
<td>8,498,762</td>
</tr>
<tr>
<td>7</td>
<td>8,234,545</td>
<td>8,365,311</td>
</tr>
<tr>
<td>7</td>
<td>8,520,733</td>
<td>8,785,457</td>
</tr>
<tr>
<td>7</td>
<td>51,681,137</td>
<td>52,117,996</td>
</tr>
<tr>
<td>8</td>
<td>37,480,171</td>
<td>37,480,825</td>
</tr>
<tr>
<td>8</td>
<td>37,480,183</td>
<td>37,480,834</td>
</tr>
<tr>
<td>8</td>
<td>61,427,491</td>
<td>62,117,211</td>
</tr>
<tr>
<td>8</td>
<td>75,557,998</td>
<td>75,699,691</td>
</tr>
<tr>
<td>8</td>
<td>82,234,280</td>
<td>82,237,190</td>
</tr>
<tr>
<td>8</td>
<td>84,332,772</td>
<td>84,408,265</td>
</tr>
<tr>
<td>8</td>
<td>90,199,458</td>
<td>90,685,026</td>
</tr>
<tr>
<td>8</td>
<td>113,033,868</td>
<td>113,467,327</td>
</tr>
<tr>
<td>8</td>
<td>125,911,054</td>
<td>125,911,344</td>
</tr>
<tr>
<td>8</td>
<td>134,425,689</td>
<td>134,428,856</td>
</tr>
<tr>
<td>15</td>
<td>92,849,263</td>
<td>93,118,873</td>
</tr>
<tr>
<td>18</td>
<td>68,384,249</td>
<td>68,761,797</td>
</tr>
<tr>
<td>18</td>
<td>70,333,020</td>
<td>70,688,598</td>
</tr>
<tr>
<td>X</td>
<td>145,041,630</td>
<td>145,841,013</td>
</tr>
</tbody>
</table>

4.2.3.7 Double minute chromosomes: results on simulated data

We applied our double minute chromosome algorithms to the simulated dataset previously described. The amplicons and the links connecting them were chosen from the Figure 4 image in the original study by Rausch et al. [33]. After applying
our algorithm to this dataset, our method correctly identified the double minute. Specifically, the framework correctly identified the amplicons (using RDXplorer) and the structural variant breakpoints that connected them (using the aforementioned NGS-based algorithms).
4.2.4 Discussion

Our results for the deletion and interchromosomal algorithms show that the usage of a combined approach (split-read and paired-read) to SV detection is advantageous over approaches that use only one signal. A method that uses only paired-reads will not only lack breakpoint prediction accuracy, but it will tend to call many more false positives. Conversely, a method that only uses split reads (such as CREST) will lose sensitivity if the variant breakpoint is not captured by many sequence reads.

For the SV prediction algorithms, our general framework of 1) paired-read detection followed by 2) breakpoint refinement using soft-clipped reads should be preferred. Furthermore, the predictions returned by each of the SV breakpoint detectors were highly concordant with the true breakpoints presented in the original study of Berger et al. [31]. Many of those breakpoints were near or within genes relevant to prostate cancer development, so our general approach is suitable for finding variants that may be clinically important to an individual.

The double minute detection algorithm correctly identified the complex event, and although we used RDXplorer for the amplicon detection, and our own SV algorithms to detect the breakpoints, the general approach is extendable to any method that predicts copy number state, and any algorithm that identifies SV breakpoints in NGS data. As stated previously, detecting complex structural variants requires methods that can uncover biologically significant associations among a group of SV breakpoints. Our method accomplishes this through the use of the amplicon graph and its auxiliary graph. Although we only created one double minute in the dataset, our method can detect distinct double minutes, since the problem of finding them
is largely (but not completely) solvable by finding the strongly connected components in the auxiliary graph. Furthermore, different double minutes can very likely be distinguished based on their copy number.
Chapter 5

Conclusion and Future Work

We have presented algorithms for detecting genomic structural variants using aCGH and NGS datasets. Our aCGH algorithm, termed CGH-Triangulator, is extremely fast, and has high sensitivity and specificity on datasets with low-to-moderate levels of noise. Our algorithms for SV detection include methods for detecting tandem repeat breakpoints, inversions, interchromosomal variants (termed “Bellerophon”), and deletions (termed “Pegasus”). The Bellerophon method can classify interchromosomal events as “non-reciprocal translocation”, “reciprocal translocation”, and “interchromosomal insertion”. Many SV programs do not attempt to classify interchromosomal breakpoints. Regarding double minute prediction, our method is the only known algorithm that attempts to detect them.

For future work, we will examine possible methods to improve the accuracy of our aCGH algorithm on noisier datasets. Possible approaches would be, for example, to triangulate every other triangle, instead of consecutive triangles. Regarding the NGS-based algorithms, we will investigate methods to filter out germline variants from our predictions, since somatically-acquired variants may be more interesting, especially in the context of disease treatment and prevention. Furthermore, since our methods rely heavily on the existence of soft-clipped reads (which may not be produced by all alignment algorithms), we will investigate possible ways to perform
breakpoint refinement without the use of soft-clipped reads. Such an extension would make our method more versatile, as it could then process read alignments from more alignment algorithms. Lastly, for the double minutes, we will apply our method to a real dataset containing real double minutes. Furthermore, we will examine ways to predict double minutes that may be “unconnected” at certain amplicon breakpoints. This could occur due because the SV algorithm fails to identify the breakpoint that connects two amplicons.
Appendix
Figure 1: Flowchart for the interchromosomal insertion prediction algorithm. For mirroring records, Bellerophon first tries to classify them as interchromosomal insertions. If it cannot, it proceeds to the balanced translocation classification step depicted in Fig. 1.
Figure 2: Flowchart for the balanced translocation prediction algorithm. Bellerophon allows for deletions and insertions of at most 1 Mb at the breakpoints. Bellerophon will pair two mirroring records whose breakpoints are closest. Although not depicted in the figure, Bellerophon allows for a maximum of two balanced translocation predictions between a pair of chromosomes. This allows our method to predict reciprocal translocations between two chromosomes that occur on both haplotypes.
Bibliography


Jeffries TW, Marth GT, Richardson PM: **Rapid whole-genome mutational profiling using next-generation sequencing technologies.** *Genome Res.* 2008, **18**:1638–1642.


[23] O’Connor C: **Human chromosome translocations and cancer.** *Nat Educ* 2008, **1**.


88


