UNIVERSITY OF CALIFORNIA, SAN DIEGO

Computational Techniques to Investigate Structural Variation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Bioinformatics and Systems Biology

by

Marcus Christopher Kinsella

Committee in charge:

Professor Vineet Bafna, Chair
Professor Kelly A. Frazer, Co-Chair
Professor Pavel A. Pevzner
Professor Jonathan Sebat
Professor Kun Zhang

2013

DEDICATION

To my Mom for her support, and to my boyfriend for his relentless encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would first like to acknowledge the wisdom and patience of my advisor, Vineet Bafna. His guidance and advice were critical for every step of my graduate work. I would like to thank my co-advisor, Kelly A. Frazer, for her valuable comments and willingness to collaborate. I would also like to acknowledge my entire committee–Pavel Pevzner, Jonathan Sebat, Kun Zhang–for sharing their advice and time. I have had the privilege to collaborate with excellent investigators: Olivier Harismendy and Masakazu Nakano at the Scripps Research Institute and the Moores Cancer Center, and Shay Zakov and Anand Patel at UC San Diego. I have also benefited from the help of many labmates and classmates who made graduate school more productive and entertaining.

Chapter 2 (with Appendix A) was published in *Bioinformatics*, Volume 27, Issue 8, pages 1068–1075, 2011, M. Kinsella, O. Harismendy, M. Nakano, K. A. Frazer, and V. Bafna, "Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs". The dissertation author was the primary investigator and author of this paper.

Chapter 3 (with Appendix B) was published in the *Journal of Computational Biology*, Volume 19, Issue 6, pages 662–678, 2012, M. Kinsella and V. Bafna, "Combinatorics of the Breakage-Fusion-Bridge Mechanism" and was presented at the 16th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2012). The dissertation author was the primary investigator and author of this paper.

Chapter 4 (with Appendix C) was published in the *Proceedings of the National Academy of Sciences of the United States of America*, 2013, S. Zakov, M. Kinsella, and V. Bafna, "Detecting Breakage-Fusion-Bridge cycles in tumor genomes—an algorithmic approach". The dissertation author was a primary co-investigator and co-author of this paper.

Chapter 5 is currently in submission, M. Kinsella, A. Patel, and V. Bafna, "Does Chromothripsis Have a Distinguishing Signature?". The dissertation author was the primary investigator and author of this paper.

All other chapters are original work of the dissertation author.

VITA

2006	Bachelor of Science in Mathematics and Biological Sciences, University of Chicago
2013	Doctor of Philosophy in Bioinformatics and Systems Biology, University of California, San Diego

PUBLICATIONS

Shay Zakov, Marcus Kinsella, and Vineet Bafna, “An algorithmic approach for breakage-fusion-bridge detection in tumor genomes.”, *Proceedings of the National Academy of Sciences of the United States of America*, 110(14), 2013

Marcus Kinsella and Vineet Bafna, “Combinatorics of the breakage-fusion-bridge mechanism.”, *Journal of Computational Biology*, 19(6), 2012

Marcus Kinsella and Vineet Bafna, “Modeling the Breakage-Fusion-Bridge Mechanism: Combinatorics and Cancer Genomics.” In Proceedings of the Sixteenth Annual Conference on Research in Computational Molecular Biology (RECOMB), pp. 148–162, 2012

Marcus Kinsella, Olivier Harismendy, Masakazu Nakano, Kelly A. Frazer, and Vineet Bafna, “Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs.”, *Bioinformatics*, 27(8), 2011

ABSTRACT OF THE DISSERTATION

Computational Techniques to Investigate Structural Variation

by

Marcus Christopher Kinsella

Doctor of Philosophy in Bioinformatics and Systems Biology

University of California, San Diego, 2013

Professor Vineet Bafna, Chair
Professor Kelly A. Frazer, Co-Chair

The importance of structural variation as a source of phenotypic variation has become more and more apparent in recent years. At the same time, tools and techniques that detect structural variation using high-throughput data have proliferated. These trends have spurred interest in making increasingly sophisticated inferences about structural variation, including identifying complex or difficult to observe variants and elucidating the biological mechanisms that produce structural variants.

Here, we identify several challenging problems in the investigation of structural variation and discuss computational techniques that solve them. First, we examine the discovery of fusion genes in the transcriptome using paired-end reads, a task complicated by reads that map to multiple locations in the genome. Earlier methods ignored these reads to control false discoveries. We demonstrate a method to resolve these ambiguous mappings and increase the sensitivity of fusion gene detection. Second, we investigate whether the breakage-fusion-bridge mechanism leaves a reliable footprint in high-throughput data, a question that had largely been addressed using ad hoc analyses. Using novel algorithms and simulation, we identify the surprisingly limited circumstances when the presence breakage-fusion-bridge can be inferred. Finally, we examine evidence for the phenomenon known as chromothripsis, the shattering and reannealing of chromosomes. We show that there are alternative hypotheses that can account for the structural variation patterns that form the currently proposed signature of chromothripsis.

Chapter 1

Introduction

1.1 The Scale of Genetic Variation

Genetic variation in humans occurs over a broad range of scales. At one extreme, gains or losses of entire chromosomes have been observed and known to cause disease for over five decades [42]. At the other, more recent efforts have cataloged and, to some extent, characterized millions of variants of single nucleotides [22].

Between these two extremes lie what are termed "structural variations." These variations alter hundreds to millions of nucleotides at once and consist of insertions, deletions, inversions, translocations, and other more exotic rearrangements. Recent studies have revealed that structural variations are both quite prevalent and contribute significantly to phenotypic variation [67, 33, 35]. Indeed, at least 12% of the human genome is subject to structural variation, meaning the majority of variable nucleotides in an individual genome are part of structural variations rather than single nucleotide variations [64]. Moreover, structural variations are implicated in numerous diseases, including schizophrenia [78, 74], autism [66], Crohn's disease [51], and psoriasis [16]. The family of diseases with perhaps the strongest association with structural variation is cancer. Of the approximately 490 genes currently known to be mutated and implicated in cancer development, over 370 are mutated via translocation, amplification, or deletion [80]. Thus, improved detection and understanding of structural variation may provide significant insight into a broad range of diseases.

1.2 Detecting Structural Variations

There are a number of experimental techniques that have been developed to provide evidence about structural variation. They vary in the type of evidence provided, and each has advantages and disadvantages. The methods used in this dissertation are described below.

Fluorescence in situ hybridization (FISH)

FISH uses fluorescently labeled probes that hybridize to nucleotide sequences from particular regions of chromosomes. Using fluorescece microscopy, these probes can be observed directly within a cell.

FISH has several advantages over other methods. FISH can give information about karyotype, so the relative location of the labeled chromosome segments with respect to the rest of the genome may be determined even in aneuploid genomes. FISH can also give the relative arrangements of labeled segments even when chromosomes are complexly rearranged. On the other hand, FISH is laborious and produces a small amount of difficult to quantify data.

Microarrays

Microarrays have multiple uses, but the application most relevant to structural variation is the detection of genomic copy number. Microarrays measure fluorescence intensities for many probes that finely cover most of the genome. These intensities can be used to infer copy numbers of different segments of the genome. Changes in copy number, amplifications and deletions, are linked to phenotypic changes and reveal some information about the structural variation within a genome. Microarrays give little information about copy-neutral variation such as inversions and translocations. It can also be difficult to infer the location where copy number changes with much precision, and copy number estimates are often rough, only showing general increases or decreases in copy number.

Sequencing

The third method that can provide information about structural variation is sequencing. Sequencing yields nucleotide substrings of the genome or transcriptome, called reads. The characteristics of these reads vary by sequencing technology, particularly in length and error profile. Sequencing can reveal information about structural variation in multiple ways. One use is similar to microarrays, except rather than fluorescence intensity, the number of reads that map to segments of the genome are measured. This metric, called "depth of coverage", can show parts of the genome that have been amplified or deleted. Another application of sequencing uses "paired-end" reads, short reads joined by an unsequenced insert. Read pairs that map in unexpected ways can reveal novel genomic adjacencies.

1.3 Algorithmic Challenges in Structural Variation Detection

While the technologies summarized abou are helpful for understanding structural variation, a gread deal of bioinformatics work must be done to convert fluorescence intensities or sequence alignments to meaningful inferences about structural variation. A number of methods have been developed that address the noisiness of biological data and the complexity of the genomes in which structural variations occur so that different kinds of structural variations can be identified. The successes of these methods have led investigators to ask increasingly sophisticated questions about structural variation. In this dissertation, we develop computational techniques and frameworks to address these sorts of questions. We identify several challenging problems related to the detection of and the mechanisms that produce structural variation and develop methods to solve them.

In Chapter 2, we examine a class of variants that are difficult to identify with existing approaches, fusions between genes with paralogs, pseudogenes, or other similar sequence elsewhere in the genome. These genes often will have few reads align to them unambiguously. We demonstrate a computational method that resolves this ambiguity to the extent possible and is able to discover novel gene fusions.

In Chapters 3 and 4, we turn to an investigation of a mechanism that produces large structural variations, the breakage-fusion-bridge mechanism. While breakage-fusion-bridge was discovered many decades ago, its detection has largely relied on cytogenetic evidence. More recently, some investigators have proposed methods to detect breakage-fusion-bridge using high-throughput data. Using simulations and novel algorithms, we show that some variation patterns that at first glance appear to be strong evidence for breakage-fusion-bridge are in fact weak evidence. We examine how varying assumptions about breakage-fusion-bridge and genome rearrangement affect inferences from high-throughput data.

In Chapter 5, examine a newly proposed phenomenon called chromothripsis that describes the shattering of a whole chromosome or chromosome segment. The segments then anneal back together in a random order, creating a complex set of breakpoints but only two copy number states.

Chapter 2

Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs

2.1 Introduction

The discovery of chimeric transcripts emerging from different and potentially distant genes has introduced another layer of complexity to the genome [23]. Additionally, the importance of fusion transcripts in the genesis and progression of cancer is becoming increasingly apparent [54, 81, 59]. Fusion transcripts may be the product of *trans*-splicing, the joining of two different transcripts emerging from distinct and often distant genes. This is especially common among lower eukaryotes [73, 40] where *trans*-splicing is part of normal transcript processing [62]. However, *trans*-splicing has also been observed in higher eukaryotes, including humans [31]. Additionally, fusions may be produced by adjacent genes yielding a single, joined RNA product, creating a read-through transcript [1]. Fusion transcripts can also result from genomic rearrangement that brings together two once distant regions of the genome. Probably the best known example of this type of fusion is BCR-ABL1, a product of a chromosomal translocation [70] found in many hematologic cancers and a successful drug target [19]. In addition, a growing list of fusion genes are being found in both hematologic and solid tumors that are the product of genomic lesions or *trans*-splicing [20]. Thus, the study of fusion transcripts has implications clinically as well for our basic understanding of the genome.

The development of high-throughput sequencing methods such as RNA-Seq [79] has offered an opportunity to hasten a fuller characterization of the transcriptome [10], including the identification of fusion transcripts. Maher *et al.* demonstrated the potential of the technology by applying transcriptome sequencing to several tumors and cancer cell lines [49, 48]. Using two different sequencing protocols, they were able to detect known fusions such as TMPRSS2-ERG [76] in a prostate cancer cell line and BCR-ABL1 in a leukemia cell line. Additionally, they identified and experimentally confirmed multiple previously-unidentified fusions. Later, Berger *et al.* carried out similar work on the melanoma transcriptome, finding 11 novel fusions [5].

Alongside these biological discoveries has been the development of computational tools and frameworks for the detection of fusion transcripts from RNA-Seq data. Ameur *et al.* developed a method for joining partial alignments of single RNA-Seq reads to find splice junctions and gene fusions[2]. Upon application of the method to a public dataset, they found hundreds of examples of transcripts that apparently spanned different chromosomes but were doubtful that many were genuine fusion genes. Hu *et al.* created a probabilistic method for aligning RNA-Seq read pairs that uses expectation maximization to find maximum-likelihood alignments[32]. They showed that paired-end reads better cover splice junctions than single reads and that their method can reliably identify splice junctions. Then, by augmenting their approach with long single reads, they were able to identify 18 gene fusions in two cancer cell lines.

Common to all of these efforts has been the requirement that a fusion transcript be supported by reads that map uniquely to the genome or transcriptome. Maher and colleagues required single best-hit mappings to the genome or mapped short, 36 nt Illumina paired-end reads to sequences derived from ~230 nt Roche 454 reads. Berger *et al.* required paired-end reads to map uniquely and at least one end of a read to unambiguously map to a junction between exons. Ameur *et al.* required each partial alignment for each read to be unique. Hu *et al.* considered fusion discovery with short paired-end reads infeasible and found putative fusions with uniquely-mapping 75 nt single reads. These strategies highlight a key difficulty in the analysis of transcriptome sequencing data:

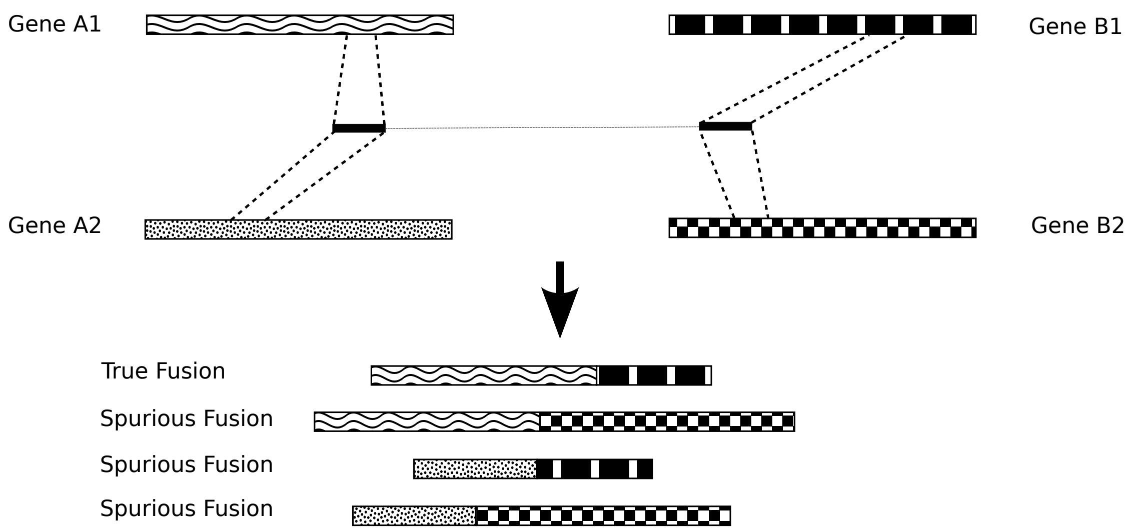


Figure 2.1: A read pair that maps to a fusion between genes A1 and B1 may also map to homologous genes, leading either to spurious fusion candidates or the elimination of read pairs supporting a true fusion from consideration.

the transcriptome is filled with repetitive and similar sequences, and many reads cannot be unambiguously mapped to a reference. Some of the repetitiveness is attributable to known repeat families such as the Alu repeat sequence, which can be found both in 5'- and 3'-UTRs as well as occasionally in coding sequence [83]. Additionally, many genes are part of gene families or have paralogs or expressed pseudogenes and thus share sequence homology with other parts of the transcriptome. Reads mapping to these genes or regions of these genes will often map well to other loci.

Ambiguously mapped reads are a concern for all transcriptome sequencing analyses and have previously been addressed by discarding them [11] or by proportionately allocating them over the different positions to which they map [56, 21]. However, this issue becomes more prominent for gene fusions because combinations of mappings are considered. Consider for example, a fusion between a pair of genes, A1 and B1. It is possible that a read pair that maps to this fusion will also map to paralogs of each gene, say A2 and B2. If all of these mappings were accepted as true, then three spurious fusions would be called (Figure 2.1). If the read pair were discarded because of its ambiguous mappings, evidence for the true fusion would be disregarded. As we detail below, our simulations indicate that these ambiguously mapping reads are present in up to 30% of possible gene fusions, underscoring the significance of the problem.

In this chapter, we propose a method to discover fusion transcripts that exploits ambiguously mapping RNA-Seq read pairs, does not require additional long, 75 nt or greater, single read sequencing, and decreases the occurrence of mapping artifacts. We begin by mapping read pairs to the transcriptome independently without imposing any unique-mappability criterion. We then find pairs which do not map to the same gene and build a set of possible gene fusions from the mappings of each read. Next, we employ a generative model of RNA-Seq data that utilizes mapping qualities and insert size distributions to resolve any ambiguous mappings. After the convergence of the expectation maximization technique used to find maximum-likelihood transcript abundances, we perform a final partial expectation step for the discordantly mapping read pairs to find optimal fusion assignments for pairs that span fusion junctions. In this way, rather than discarding ambiguously mapping read pairs or allowing them to overstate the number of fusions present, we find the best supported fusions by using the mappings of all the reads in the dataset, the quality of those mappings, and the implied insert sizes of read pairs that span a fusion site. This allows our method to more sensitively detect gene fusions than if ambiguously mapping read pairs were discarded.

We have implemented our method on simulated data generated from fusions between genes with very high similarity to other genes to demonstrate that our method can resolve the ambiguous mappings to find the correct fusions when it is possible to do so. We then implemented it on reads derived from neoplastic and hyperplastic prostate tissue and recovered the known TMPRSS2-ERG fusion along with several read-through fusions without finding many spurious, poorly supported fusions as a result of allowing reads to have many mappings. Finally, using publicly available data from several melanoma tumors and cell lines, we find fusion events that would not be detectable without allowing for multimapping reads that span the fusion site.

2.2 Methods

2.2.1 Discovery of Putative Fusions

The first step of our method is to map each read of a pair independently. We use Bowtie [41] in single-end mode to perform this mapping against a database of RefSeq transcripts [60] that have been prepended with 50 nt of upstream sequence and appended with string of adenines to account for variation in transcription start site and polyadenylation, respectively. Filtering the mapping results yields a set of read pairs that only map discordantly to different genes. Then, to decrease the possibility of generating inauthentic fusions as a result of SNPs or mapping or annotation errors, we map these discordant read pairs to the genome and transcriptome, and we greatly relax the stringency of reported mappings and allow for many mappings to be reported for each read. For the experiments in this study, we use the Bowtie flags -l 22 -e 350 -y -a -m 5000. These flags cause Bowtie to report all mappings for each read, to try as hard as possible to find valid mappings, and to suppress mappings with more than two mismatches in the first 22 bases, summed quality values at all mismatched positions greater than 350, or mappings from reads with more than 5000 reportable mappings. With these less stringent mappings, we check if each pair of reads both map within the genomic bounds of a known gene or within 10 kilobases of each other in a region of the genome with no annotated genes. This filtering step decreases the possibility of events such as retained introns or unannotated transcripts being mistakenly called as gene fusions.

After these filtering steps, we consider each pair of genes to which at least two read pairs map discordantly with fewer than 3 mismatches. Our aim is to determine which exons from each gene should comprise a putative fusion transcript. Combinations of exons are required to satisfy three conditions. First, all exons upstream of the junction site in the upstream gene isoform and all exons downstream of the junction site in the downstream gene isoform must be included. So, in Figure 2.2 fusion 4, exon 4 from gene A could not be included without also including exon 3. Second, all exons to which a read maps must be included. For example, in Figure 2.2, exon 1 and exon 2 from gene A must be included because reads map to them. Third, the implied insert size of any read pair should not be unreasonably large given the known insert size used for sequencing. For example, in Figure 2.2 fusion 4, the insert size of read pair 3 implied by the inclusion of exons 3 and 4 from gene A may be too large. To decrease the sensitivity of otherwise acceptable exon combinations to occasional abnormally long insert sizes, we allow one tenth of read pairs to violate this third criterion. While there are certain types of fusions that would not meet these criteria, say a fusion with multiple, similarly expressed isoforms that vary near the fusion site, we find that these criteria effectively eliminate many spurious fusions without losing sensitivity to *bona fide* ones.

Usually, there are multiple combinations of exons from each gene pair that satisfy the above criteria. To enumerate them efficiently, we find every pair of RefSeq isoforms from each gene pair that is supported by at least two discordantly mapping read pairs. For each isoform pair, we build a directed graph of their exon structures augmented with edges that connect each exon in the upstream isoform to each exon in the downstream isoform (Figure 2.3). Then, we search for paths from the beginning of the upstream isoform to the end of the downstream isoform by implementing a depth-limited search:

```
Algorithm DLS(node, path, reads)
Input: A node representing an exon, a path through the exon graph, and a set of reads
    mapping to the exons.
Output: Paths through the exon graph that satisfy the above criteria.
1.  if node is in downstream gene and not all reads marked open or closed
2.    then return
3.  if node is in upstream gene
4.    then for read that maps to node
5.              mark read as open
6.    else  for read that maps to node
7.              mark read as closed
8.  for read marked as open
9.          add length of node's exon to implied read insert
10. if (count of read pairs with insert> max_insert_size) > .1*(count of reads)
11.   then return
12. if node is sink node
13.   then output path
14.   else  for neighbors of node
15.             DLS(neighbor, path + node, reads)
```

DLS is initially called with the root node *S*, an empty path, and the set of discordantly mapping read pairs for the isoform pair. It then proceeds through the graph in a depth-first fashion. At each node it checks if there are reads mapping to that node and opens or closes each read pair appropriately, keeping track of the state of each pair independently. If a read maps to a splice junction, the inner boundary of the mapping is used to determine the exon to which it maps. When a read maps to an exon, only the appropriate portion of the exon's length is added to the implied insert size in line 9. The directed edges of the graph ensure that the first criterion above is met. The second and third criteria are ensured explicitly in lines 1 and 2 and lines 10 and 11, respectively. Since the depth of any search path is limited, this procedure can efficiently discover fusions that meet our desired criteria. In addition, to better facilitate the detection of read-through transcripts, the 3' exon of the upstream gene and the 5' exon of the downstream gene do not contribute to the reads' implied insert sizes. This follows from our observation that these exons often appear truncated in read-through fusions. Finally, since different isoforms of the same gene mostly contain the same exons, duplicate exon sets can be generated by calling *DLS* on different isoforms. These duplicates are removed before proceeding to the next step.

2.2.2 Mapping to Augmented Reference

After the set of putative fusions are generated, the sequence for each is generated and added to the original set of transcripts from RefSeq. Then, the read pairs are mapped to this augmented reference. Unlike the previous mapping, Bowtie is used in paired-end mode and the default mapping stringencies are used except that up to 1500 possible mappings for each paired-end read are allowed. While the addition of the putative fusion sequences may result in the addition of thousands of additional transcripts to the reference, the total amount of sequence in the augmented reference remains smaller than the genome, and the mapping can still be carried out on a standard desktop computer. After mapping, we proceed, as discussed below, to ranking fusions based on coverage.

2.2.3 Model of Paired-End RNA-Seq Data

We extend the generative model of [43] to develop a probabilistic model for generating read pairs (Figure 2.4). We reason that a read pair is generated in four steps.

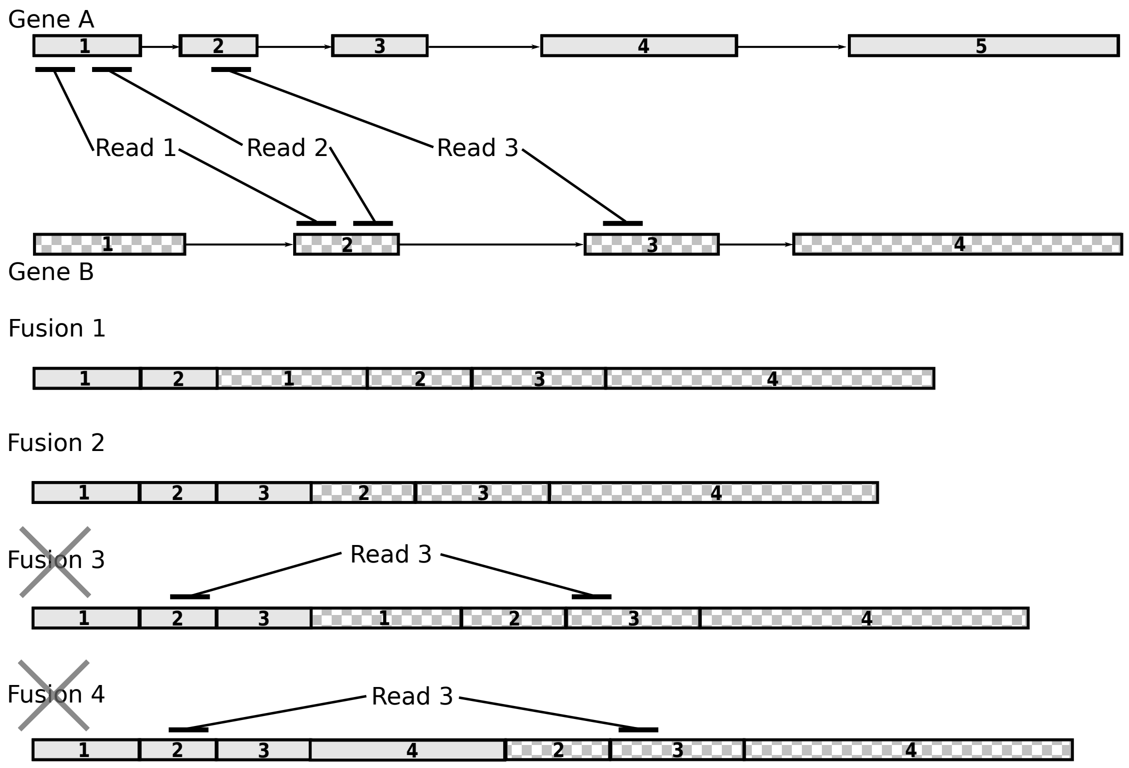


Figure 2.2: Creating fusion genes from discordantly mapping mate pairs. Three mate pairs map to two different gene isoforms. Fusion 1 and Fusion 2 include all the exons in either isoform covered by reads. Fusions 3 and 4 also do, but they are rejected because the implied insert size for Read 3 is too large.

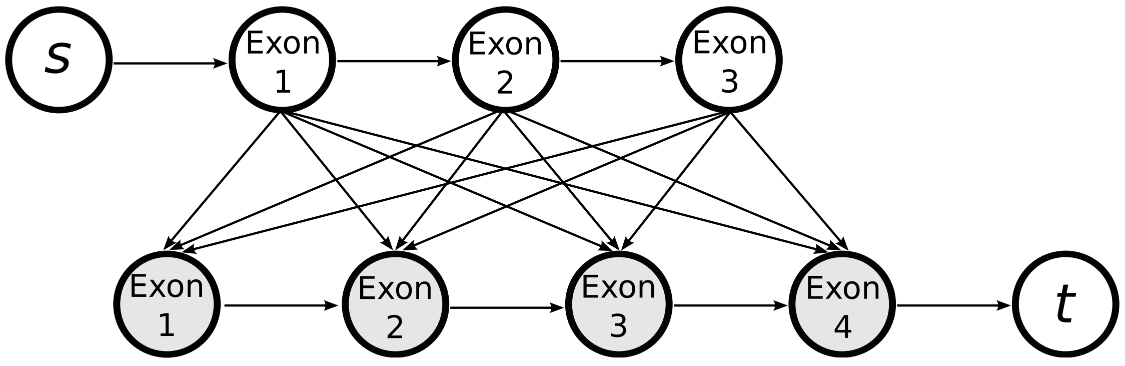


Figure 2.3: To nominate potential fusion transcripts, we build a graph from the exons of each gene isoform in the pair. By adding edges from the upstream transcript to the downstream transcript, we find paths that account for all read pairs mapped to the fusion and that respect an upper bound for the insert size of the read pairs.

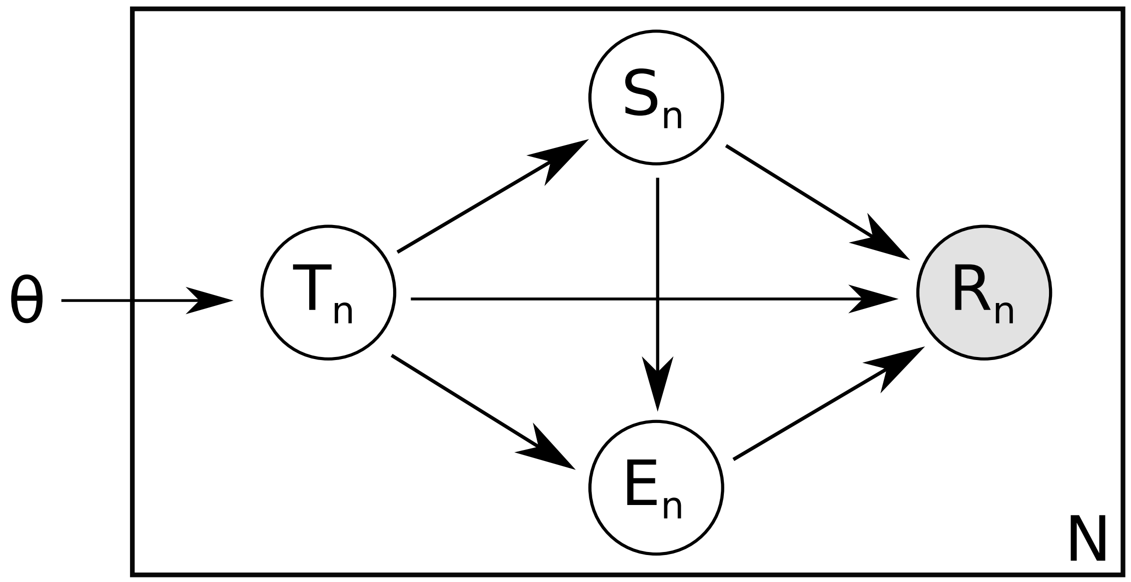


Figure 2.4: The graphical model of RNA-Seq read pairs. Transcript abundance, transcript choice, starting position, ending position, and observed read are represented by θ, T, S, E, and R, respectively.

First the transcript from which the pair will come, t_n, is chosen. Then the starting point for the upstream read, s_n, within that transcript is chosen; then the end point for the downstream read, e_n, is chosen. Finally, errors are introduced and the final read pair is observed. As we only observe reads, we can consider transcript choice, starting position, ending position, and read error to be hidden variables. The likelihood of a collection of read pairs, and specific values of the hidden variables can be expressed as a function of the true transcript nucleotide abundances:

$$P(\mathbf{R},\mathbf{T},\mathbf{S},\mathbf{E}|\theta) = \prod_{n=1}^{N} P(t_n|\theta)P(s_n|t_n)P(e_n|s_n,t_n)P(r_n|e_n,s_n,t_n)$$

Each term in this equation can be calculated in a straightforward way. The probability of a transcript t being chosen is the relative nucleotide abundance of that transcript, that is, the fraction of all nucleotides that are part of that transcript. Thus, $P(t_n|\theta) = \theta_t$. Assuming that each base within a transcript is equally likely to be the starting point of the upstream read, the probability of a particular starting point is the inverse of the length of the transcript ℓ_t: $P(s_n|t_n) = \ell_t^{-1}$. The choice of the ending point depends on the distribution of insert sizes used for sequencing and the starting point. We use $d(|s_n - e_n|)$

to indicate the value of the insert size distribution for the distance between the start and end points, which we empirically determine from the read pairs that map concordantly. Finally, the probability of a read being observed from a given transcriptomic locus can be calculated using matches and mismatches between the read sequence and the reference transcriptome and the quality values of the bases in the read [44]. We denote this probability as $\varepsilon(r_n, t_n, s_n, e_n)$.

To expand the probability distribution to N read pairs, we take the product of values for individual reads.

$$P(\mathbf{R}, \mathbf{T}, \mathbf{S}, \mathbf{E}|\theta) = \prod_{n=1}^{N} \theta_{t,n} \ell_{t,n}^{-1} d(|s_n - e_n|) \varepsilon(r_n, t_n, s_n, e_n)$$

Finally, the probability of our observed variable, the read pairs, given the transcript abundances can be calculated by summing over the values of the hidden variables.

$$P(\mathbf{R}|\theta) = \prod_{n=1}^{N} \sum_{t\prime, s\prime, e\prime} \theta_{t\prime,n} \ell_{t\prime,n}^{-1} d(|s\prime_n - e\prime_n|) \varepsilon(r_n, t\prime_n, s\prime_n, e\prime_n)$$

We seek to find the set of transcript abundances, θ, that maximizes this probability by applying expectation maximization to the results of the paired-end mapping to the reference augmented with the putative fusions.

2.2.4 Expectation Maximization

For consistency, we use notation similar to that used by Li *et al*. Let $Z_{nijk} = 1$ if $(t_n, s_n, e_n) = (i, j, k)$. Then, as the first step of the EM algorithm, we find the expected values of Z_{nijk} given the observed reads and the current estimate of θ.

$$E_{Z|R,\theta^{(t)}}[Z_{nijk}] = \frac{\theta_i^{(t)} \ell_i^{-1} d(|j-k|) \varepsilon(n, i, j, k)}{\sum_{i\prime, j\prime, k\prime} \theta_{i\prime}^{(t)} \ell_{i\prime}^{-1} d(|j\prime - k\prime|) \varepsilon(n, i\prime, j\prime, k\prime)}$$

Then, the E-step consists of calculating the log-likelihood weighted by these values.

$$Q(\theta|\theta^{(t)}) = \sum_{n,i,j,k} E_{Z|R,\theta^{(t)}}[Z_{nijk}] \log(\theta_i \ell_i^{-1} d(|j-k|) \varepsilon(n, i, j, k))$$

The values for $\theta^{(t+1)}$ are then found by finding the θ that maximizes this function subject to the constraint $\sum_{i=1}^{M} \theta_i = 1$ using Lagrange multipliers.

$$\Lambda = Q(\theta|\theta^{(t)}) + \lambda(\sum_{i=1}^{M} \theta_i - 1)$$
$$\frac{\partial \Lambda}{\partial \theta_i} = \sum_{n,j,k} \frac{E_{Z|R,\theta^{(t)}}[Z_{nijk}]}{\theta_i} + \lambda$$

Equating all of these terms to zero, we have

$$\begin{aligned}\theta_i^{(t+1)} &= \frac{\sum_{n,j,k} E_{Z|R,\theta^{(t)}}[Z_{nijk}]}{\sum_{n,i,j,k} E_{Z|R,\theta^{(t)}}[Z_{nijk}]} \\ &= \frac{1}{N} \sum_{n,j,k} E_{Z|R,\theta^{(t)}}[Z_{nijk}]\end{aligned}$$

This procedure is repeated until convergence. We make the probability calculations tractable by only considering, for each read, the values of t, s, and e reported by short read mapping software and assuming the probability of the read coming from any other position to be zero.

2.2.5 Calculating Mappings to Fusion Junctions

After convergence of the expectation-maximization algorithm, we have an estimate of the maximum-likelihood abundances for each transcript, including all of the putative fusion transcripts. These abundances reflect the resolution of read mapping ambiguity, as demonstrated by the successful elimination of many spurious fusions in the results below. However, they do not yet account for potential unevenness of coverage across a given transcript. In particular, they can be confounded by a fusion transcript with high coverage everywhere but the fusion site. To illustrate this issue, consider the situation illustrated in Figure 2.5. We have three reference transcripts: Gene A, Gene B, and a fusion gene created by concatenating Gene A and Gene B. We also have three sets of read pairs: N_A pairs that map to Gene A and the fusion gene, N_B pairs that map to Gene B and the fusion gene, and N_F pairs that only map to the fusion gene. For simplicity, assume that the values of $\varepsilon(r_n, t_n, s_n, e_n) = 1$ and $d(|s_n - e_n|) = 1$ for each mapping of each read pair and the length of both Gene A and Gene B is 1. Then, the probability of the observed data is

$$P(\mathbf{R}|\theta) = (\theta_A + \frac{1}{2}\theta_F)^{N_A}(\theta_B + \frac{1}{2}\theta_F)^{N_B}(\frac{1}{2}\theta_F)^{N_F}$$

If we further assume that $N_A = N_B$ and therefore $\theta_A = \theta_B$, and use the fact that the sum of the transcript abundances must be 1, we have that $\theta_A = \frac{1-\theta_F}{2}$. Then, the probability of the observed data becomes

$$P(\mathbf{R}|\theta) = (1)^{N_A}(1)^{N_B}(\frac{1}{2}\theta_F)^{N_F}$$

This expression is maximized by setting θ_F to 1, which sets θ_A and θ_B to zero. So, if there is a single read pair that spans the fusion site in this scenario, all abundance is transferred to the fusion transcript regardless of how large N_A and N_B may be in relation to N_F. While this example has been rather stringently defined for sake of demonstration, a similar situation occurs whenever $N_F > 0$ and $N_A >> N_F$ or $N_B >> N_F$: an unreasonable abundance is assigned to the fusion transcript based on reads that do not map to the fusion site. In the context of seeking fusions, this means a fusion between highly expressed genes supported by a single read pair, perhaps an experimental artifact, will dominate other putative fusions in abundance. To avoid this, rather than simply using the maximum-likelihood abundances, we calculate the sum of the expected values of Z_{nijk} for each fusion transcript i for read pairs that span the fusion junction to get a probabilistically weighted count of reads supporting the fusion, C_i.

$$C_i = \sum_{n,j,k} E_{Z|R,\theta^{(final)}}[Z_{nijk}] \text{ for n} \in \text{pairs spanning junction}$$

This retains the ambiguity resolution described above but focuses the abundance estimates on fusions.

As a final filtering step to eliminate experimental artifacts, we find the mean physical coverage, that is, the coverage counting both the reads and the insert, for the upstream gene and for the downstream gene in the fusion separately and compare each of them to the physical coverage at the fusion site. If coverage at the fusion site is less than one twentieth of the upstream and downstream coverage, we discard the fusion as a probable artifact based on the same reasoning discussed above. We also discard fusions where all reads have the sequence of an RNA component of the spliceosome, U1 through U6, as these are likely produced artifactually as well.

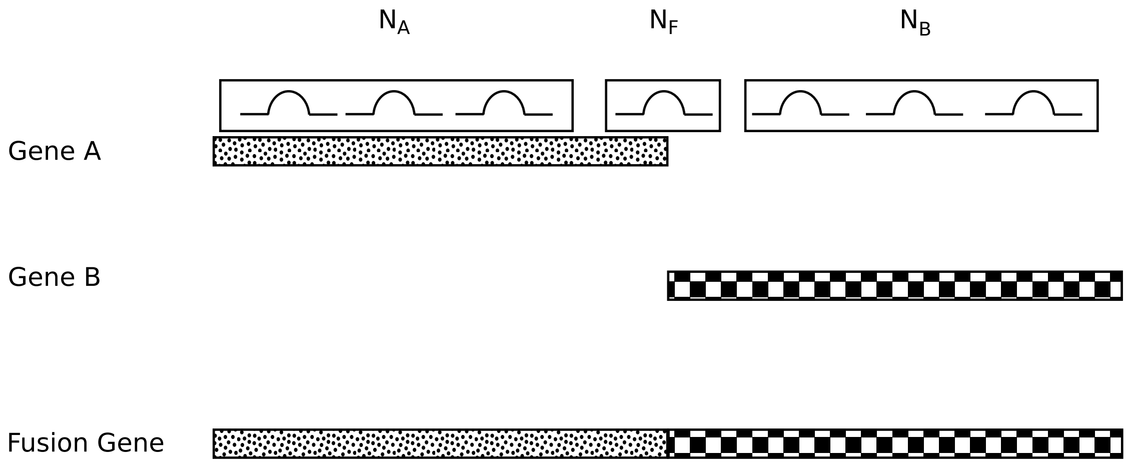


Figure 2.5: In this simplified situation, maximizing the likelihood function would set the abundance of the fusion gene to 1 regardless of the relationship between N_A, N_B, and N_F.

2.3 Results

2.3.1 Fusion Transcripts Generate Ambiguous Reads

To quantify the prevalence of ambiguously mapping read pairs and the extent to which discarding them would impact fusion discovery, we simulated gene fusions by randomly selecting a pair of transcripts from RefSeq and the exon within each transcript that would serve as the fusion breakpoint. For each fusion, we generated, with random errors based on quality scores from an existing dataset, the full set of read pairs that could span it given a constant insert size. We then mapped each of these reads and tabulated the number of read pairs with unique mappings that satisfy default Bowtie mapping criteria [41]. We repeated this for several read lengths, generating 100,000 simulated fusions for each read length, while keeping the insert between the two reads at 200 nt.

For each read length, we calculated the fraction of partially ambiguous fusions and totally ambiguous fusions, that is, fusions where some, but not all, of the reads supporting them mapped ambiguously and fusions that only generated ambiguously mapping read pairs. As expected, the fraction of ambiguous fusions declined as read length increased. At a read length of 50 nt, nearly one in twenty fusions would only be detectable via ambiguously mapping read pairs (Table 2.1, A.1). Even at a read length of

100 nt, over a tenth of all fusions were able to generate an ambiguously mapping read pair. These results suggest that even as read lengths increase, a significant portion of fusions remain difficult to detect if read pairs are required to map unambiguously.

Table 2.1: The fraction of totally and partially ambiguous fusions for a range of read lengths.

Read Length	**% Partially Ambiguous Fusions**	**% Totally Ambiguous Fusions**
30	30.3	5.7
35	22.4	5.5
40	17.5	5.1
45	14.9	4.8
50	13.4	4.5
75	9.4	3.7
100	7.9	2.9

2.3.2 Resolving Ambiguous Simulated Fusions

To demonstrate the capability of our method to find gene fusions between highly repetitive regions of the transcriptome using multimapping read pairs, we simulated five fusion genes, outlined in Table 2.2, derived from possible fusions between genes that share homology with other parts of the transcriptome. Then, 10,000 pairs of 40 nt reads were generated from these five fusions using MAQ [44] in simulate mode with insert size set to 200 nt. Sequencing errors and quality values were modeled from an existing dataset, and the MAQ simulation code was modified to produce a distribution of different expression levels for each transcript so performance over a range of coverage levels could be examined. As a comparison, the coverage levels used in the simulation would correspond to a range of approximately 8 FPKM for MAGED4-MBD3L2 to 80 FPKM for FOXO3-EIF3CL in a 20 M read pair sequencing experiment. Thus, the simulated coverages provide a reasonable range on which to evaluate the performance of our method.

Mapping the 10,000 read pairs to RefSeq transcripts yielded 395 that mapped only discordantly. As expected, all of these discordantly mapping pairs mapped to multiple genomic loci and thus suggested multiple fusion candidates. Each discordant read

Table 2.2: Simulated fusions.

Gene 1	Gene 2	Pair Count	Pairs Spanning Fusion
FOXO3	EIF3CL	7152	281
PSG2	PHB	1324	117
FRG1	USP6	803	47
SMN2	CSAG1	434	78
MAGED4	MBD3L2	286	34

pair mapped, on average, to seven different pairs of genes, and in some cases mapped to as many as 22. The total number of fusion genes that would be nominated by naïvely accepting all discordant mappings was 56 (Table A.2).

Applying the filtering and fusion discovery process described in the Methods Section 2.1 yielded 252 putative fusion transcripts. The high number reflects both the multiple gene pairs to which the discordant read pairs mapped and the multiple sets of exons from each gene pair that could be consistent with the discordant mappings.

After allowing the estimate of the maximum-likelihood transcript abundances to converge, only 12 of the 252 nominated fusion transcripts had at least two read pairs assigned to its junction site. Those 12 transcripts represent 7 potential fusion genes (Table 2.3). All five of the fusions from which the data were generated are included in the results. In addition, two spurious fusions are reported. The results include a fusion between FOXO3 and EIF3C in addition to the true fusion between FOXO3 and EIF3CL. However, this is not a failing of the algorithm. The sequences of EIF3C and EIF3CL are very nearly identical; depending which isoform of each gene is considered, they differ at most by several bases at the end of their 3' exons. So, every read that maps to the fusion of FOXO3 and EIF3CL also maps to the fusion of FOXO3 and EIF3C. Rather than discard these reads, the algorithm simply preserved this unresolvable uncertainty and divided them between the two fusions according to values obtained from the probabilistic model. Similarly, SMN1 and SMN2 are nearly indistinguishable. Thus, using only ambiguously mapping read pairs, our method recovered the five true fusions, eliminated 49 spurious ones, and retained two fusions that are indistinguishable from true fusions.

Table 2.3: Sum of expected values of Z_{nijk} for read pairs supporting each fusion after maximum-likelihood transcript abundance estimation.

Upstream Partner	Downstream Partner	Supporting Read Pairs
FOXO3	EIF3C	180.3
PSG2	PHB	117.0
FOXO3	EIF3CL	100.6
SMN1	CSAG1	56.6
FRG1	USP6	46.9
MAGED4	MBD3L2	34.0
SMN2	CSAG1	21.4

2.3.3 Application to a Prostate Tissue Transcriptome Data

We applied our method to two datasets derived from tissue resected from an individual with prostate cancer. The first dataset consisted of 18,027,834 pairs of 40 nt reads from neoplastic tissue. The second was 21,978,463 read pairs from adjacent hyperplastic tissue. Of the neoplasia read pairs, 18,177 had only discordant mappings and mapped to 127,102 gene pairs. Of the hyperplasia read pairs, 24,569 had only discordant mappings and mapped to 266,571 gene pairs. Application of the filtering and fusion discovery process described above yielded 887 and 746 putative fusion transcripts for neoplasia and hyperplasia, respectively. After estimating transcript abundances, only 15 fusion transcripts from the neoplasia data had at least two reads assigned to its junction site (Table 2.4). The top result, a fusion between TMPRSS2 and ERG, is a known recurrent fusion in prostate cancer [76]. A novel fusion between GRHL2 and SNTG1 was also reported. These genes lie about 50 megabases apart on chromosome 8. Intriguingly, there is a short sequence shared by both sequences at the site of the fusion (Figure A.2) , potentially providing a clue to the origin of the chimera [45]. The remaining results were read-through transcripts present in existing EST databases [4].

In sharp contrast to the neoplasia results, the hyperplasia data showed no evidence of a fusion between TMPRSS2 and ERG (Table 2.5). This is consistent with the central role the TMPRSS2-ERG fusion is suspected to play in the progression of prostate cancer [82]. Beyond this critical difference, the results largely mirrored those from neoplasia. There was one novel read-through transcript reported, RPL7-LOC100130301,

Table 2.4: Prostate neoplasia fusions with sum of expected Z_{nijk} values.

Upstream Partner	**Downstream Partner**	**Supporting Read Pairs**
TMPRSS2	ERG	49.0
AZGP1	GJC3	28.0
TTY14	NCRNA00185	8.0
LOC728606	KCTD1	4.0
ZNF649	ZNF577	3.0
SMA4	GTF2H2B	2.5
LOC100134368	NME4	2.0
SYNJ2BP	COX16	2.0
SMG5	PAQR6	2.0
PRKAA1	TTC33	2.0
LOC401588	CHST7	2.0
HARS2	ZMAT2	2.0
UQCRQ	LEAP2	2.0
GRHL2	SNTG1	2.0
KLK4	KLKP1	2.0

and multiple previously reported read-throughs: AZGP1-GJC3, SPINT2-C19orf33, DHRS1-RABGGTA, TMEM203-C9orf75, and IRF6-C1orf74. The large number of potential fusions suggested by a naïve examination of discordant reads, over 100,000 in each dataset, underscores the complexity of the transcriptome and the often muddled nature of experimentally-derived transcriptomic sequencing data. We were gratified that our method was able to discard nearly all of these inauthentic fusions while retaining those of biological importance.

Table 2.5: Prostate hyperplasia fusions with sums of expected Z_{nijk} values.

Upstream Partner	**Downstream Partner**	**Supporting Read Pairs**
AZGP1	GJC3	54.0
SPINT2	C19orf33	6.8
RPL7	LOC100130301	3.0
TMEM203	C9orf75	3.0
DHRS1	RABGGTA	3.0
IRF6	C1orf74	2.0

2.3.4 Discovery of Novel Ambiguous Fusions

To demonstrate the ability of our method to make new discoveries, we analyzed two publicly available datasets. The first was transcriptome sequencing of a set of melanoma tumors and cell lines originally published by [5]. The second was sequencing of Stratagene's Universal Human Reference RNA (UHR), a reference composed of RNA from ten cell lines originally published by [8]. Analysis of these data with our method yielded numerous fusions, including all of the fusions reported by Berger and numerous fusions known to be present in UHR including BCR-ABL1, BCAS4-BCAS3, and GAS6-RASA3 (Table A.3). In addition, we found five fusion transcripts where some or all of the read pairs mapping to them also mapped to other potential fusions (Table 2.6). In each case, the ambiguity was due to genomic duplications. Some reads mapping to the MYH6 side of the HOMEZ-MYH6 fusion also mapped to MYH6's paralog, MYH7 (Figure 2.6). The remaining ambiguous fusions were due to recent segmental duplications. The fusion between CPEB1 and RPS17 was clearly a read-through, but was confounded by the presence of another copy of RPS17 in an upstream segmental duplication (Figure 2.7). KIAA1267-ARL17A was similarly made ambiguous by multiple copies of ARL17. The fusions between PPIP5K1-CATSPER2 and TRIM16L-FBXW10 were confounded by mappings to CATSPER2P1 and TRIM16-CDRT1. The sequence of each fusion is available in Section A.1. These findings confirm that additional fusions can be detected in tumors when ambiguously mapping read pairs are included in the analysis.

2.4 Discussion

In this chapter, we have demonstrated a method to use discordantly and often ambiguously mapping RNA-Seq read pairs to identify fusion transcripts. In doing so, we bring the increasingly sophisticated methods employed to estimate transcript abundance in the presence of multimapping reads to the problem of fusion discovery. In contrast to previously proposed methods for fusion identification that focus on reads that map to the junction between two genes [2], our method estimates fusion transcript abundances by considering physical coverage over the entire length of the proposed fusion. In addition,

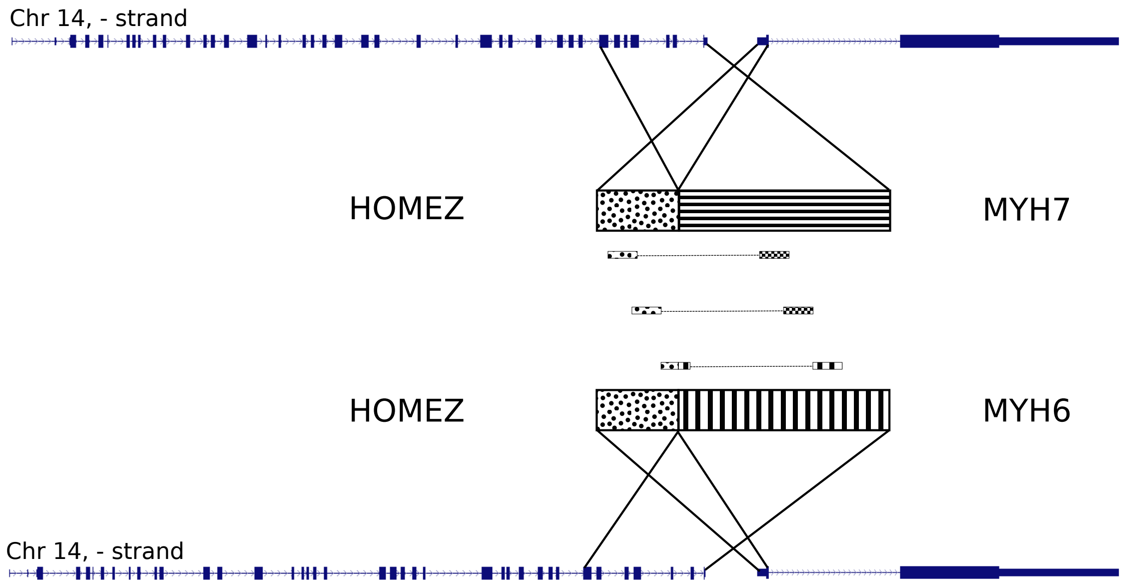


Figure 2.6: The fusion between HOMEZ and MYH6. Three mate pairs support this fusion, but two also map to a fusion between HOMEZ and MYH7.

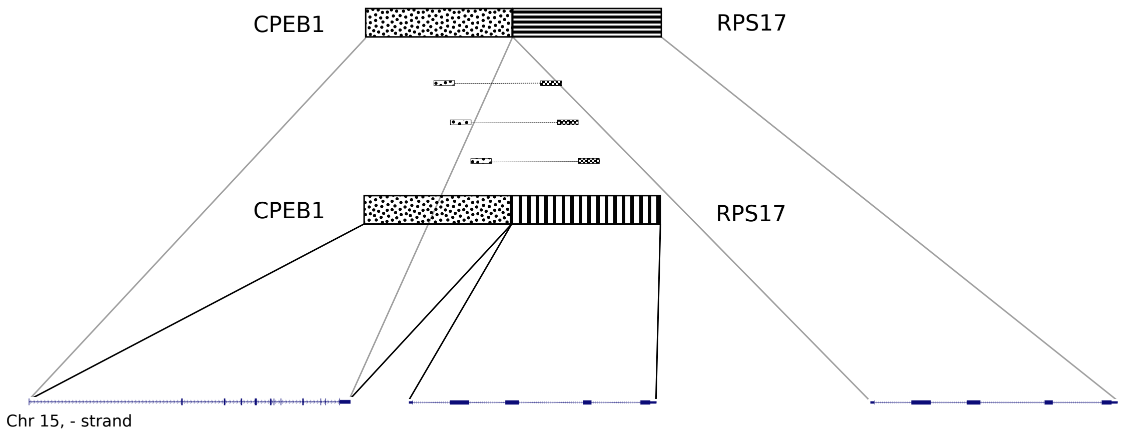


Figure 2.7: The fusion between CPEB1 and RPS17. A copy of RPS17 lies 2,000 bases downstream of CPEB1, but another copy lies 400 kilobases downstream, as well.

Table 2.6: Fusions found in previously published datasets that are either partially or completely supported by ambiguously mapping read pairs.

Fusion	Samples	Supporting Read Pairs	Ambiguous Read Pairs
HOMEZ-MYH6	UHR	3	2
KIAA1267- ARL17A	M000216	11	11
	M010403	11	11
	UHR	11	11
CPEB1-RPS17	M980409	3	3
	MeWo	5	5
PPIP5K1- CATSPER2	M010403	4	3
	M990802	17	13
TRIM16L-FBXW10	M010403	3	3

it employs several filters to minimize experimental artifacts. Finally, it does not require that any single read sequence hit the point of fusion. Instead, it uses implied insert sizes and known exon boundaries to determine the most likely point of fusion. This would be a liability if a fusion transcript contained partial exons, but reported fusions to date suggest that a vast majority of fusions do indeed involve the joining of whole exons from different genes, the breakpoints occurring in introns and the splice sites remaining unchanged [25].

Several avenues for future development are apparent from this work. Here, we chose to use RefSeq transcripts as the reference against which reads are mapped. This allowed us to avoid the issue of reads that map to splice junctions because the splice junction sequence would be contiguous in the transcript sequence. However, it prevents us from identifying transcripts that are produced by novel or aberrant splicing, which is common in cancer [61], or are significantly altered by RNA-editing [71]. It may be fruitful to combine the approach described here with methods that identify splice junctions and expressed regions of the genome *de novo* [2, 77]. Additionally, fusion transcript discovery shares many parallels with the problem of resolving genomic rearrangements, especially the challenges of repetitive sequence. The adaptation of the methods developed here to genomic sequencing may prove useful in this related field.

2.5 Acknowledgements

We thank Eric Topol for his support and the clinical team at The Scripps Translational Science Institute for sample collections.

This work was supported by the National Institutes of Health [grant numbers RO1-HG004962, 5U54HL108460, 1R21CA152613-01, CIRM DR1-01430, and CTSA-1U54RR025204] and the California Institute for Regenerative Medicine [grant number DR1-01430].

Chapter 2 (with Appendix A) was published in *Bioinformatics*, Volume 27, Issue 8, pages 1068–1075, 2011, M. Kinsella, O. Harismendy, M. Nakano, K. A. Frazer, and V. Bafna, "Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs". The dissertation author was the primary investigator and author of this paper.

Chapter 3

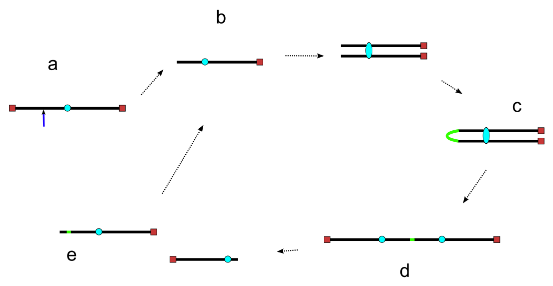
Combinatorics of the Breakage-Fusion-Bridge Mechanism

3.1 Introduction

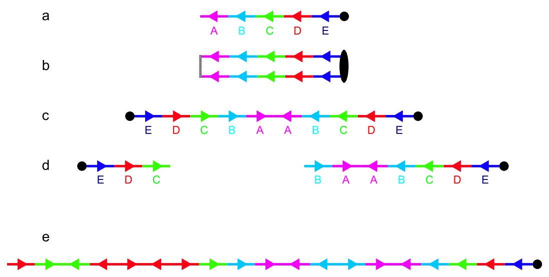
The breakage-fusion-bridge (BFB) mechanism was first proposed by Barbara McClintock in 1938 to explain observations of chromosomes in maize [52, 53]. BFB begins with a chromosome losing a telomere, perhaps through an unrepaired DNA break or through telomere shortening. As the chromosome replicates, the broken ends of each of its sister chromatids fuse together. During anaphase, as the centromeres of the chromosome migrate to opposite ends of the cell, the fused chromatids are torn apart. Each daughter cell receives a chromosome missing a telomere, and the cycle can begin again (Figure 3.1a).

When the fused chromosome is torn apart during anaphase, it likely does not tear exactly in the middle of the two centromeres. As a result, one daughter cell receives a chromosome with a terminal inverted duplication while the other receives a chromosome with a terminal deletion. After many BFB cycles, repeated inverted duplications can result in a dramatic increase in the copy number of segments of the unstable chromosome (Figure 3.1b).

BFB's ability to amplify chromosomal segments suggests a role for the mechanism in cancer. Gene amplification is common in tumors. A recent review identified 77



a. The BFB mechanism is a multiple step process. First, a chromosome loses a telomere (a). Then, the telomere-lacking chromosome (b) replicates. The sister chromatids lacking telomeres fuse together (c). During anaphase, the centromeres separate, forming a dicentric chromosome (d). As the centromeres migrate to opposite ends of the cell, the chromosome is torn apart, and each daughter cell gets a chromosome lacking a telomere (e).



b. After telomere loss (a), sister chromatid fusion (b), and centromere separation (c), the dicentric chromosome may not break in the center of the two centromeres (d). In this case, the break was between the green and cyan segments, so one daughter cell will have a a deletion of those segments while the other will have an inverted duplication. Multiple rounds of inversion and duplication can lead to amplification of chromosomal segments (e).

Figure 3.1: The Breakage Fusion Bridge mechanism.

genes whose amplification is implicated in cancer development [65]. Multiple lines of evidence indicate that BFB may be responsible for much of this amplification. Telomere dysfunction and crisis is associated with tumorigenesis [3]. Such dysfunction is consistent with the initiation and continuation of BFB cycles. Some tumors also display the cytogenetic hallmarks of BFB: chromosomes that stretch across spindle poles during anaphase, dicentric chromosomes, and homogeneously staining regions. Thus, an improved understanding of BFB may shed light on how genomic instability leads to tumor formation and progression.

Observing BFB through classical cytogenetic techniques can be difficult and only provides coarse detail. Recent studies have begun to apply modern methods to the problem of detecting BFB and elucidating its role in generating genomic aberrations. Kitada and Yamasaki performed FISH and array CGH on a lung cancer cell line and showed that the pattern of amplification and rearrangement they observed was consistent with BFB [37]. Later, Bignell *et al.* sequenced breakpoints in a breast cancer cell line and confirmed that the copy count and breakpoint patterns on 17q were consistent with a BFB model, purportedly the first demonstration of a sequence-level hallmark of BFB in human cancer [7].

These studies, among others, illustrate the promise of new methods for gaining a more complete understanding of BFB. However, making observations that are consistent with a model does not allow one to conclude that the model is correct. Moreover, without a precise definition of the model under consideration, an investigator cannot use data to refine the model and may succumb to bias when considering evidentiary support for the model.

To address these concerns, we present perhaps the first formal description of the BFB mechanism. We use this formalization to consider the range of amplification patterns that can be produced by BFB. The underlying algorithmic problems are challenging and of unknown complexity. We develop heuristic algorithms and rules to determine if a given pattern is consistent with BFB, based on, among other observations, a tight connection between BFB patterns and trees with certain symmetries. The methods make the problems tractable for practical instances.

Using these methods, we show that BFB-associated amplification patterns are common. In fact, if some imprecision is allowed, a majority of possible patterns are consistent with BFB. This suggests that observing that an amplification pattern could have been produced by BFB is not conclusive evidence that BFB produced the amplification.

3.2 Formalizing the BFB Schedule

Our first task is to describe a model of the breakage-fusion-bridge mechanism that is both consistent with its biological features and is amenable to computational techniques. We begin by considering a chromosome arm that has lost its telomere. Suppose we label potentially unequally sized intervals along this chromosome arm A,B,C,... from the telomeric end to the centromere. Then, we trace the fate of these intervals through a BFB cycle.

For illustration, suppose that the chromosome arm, after loss of the telomere, is composed of five intervals, ABCDE. Then, the chromosome is analogous to Figure

3.1b(a) where A,B,C,D,E correspond to the magenta, cyan, green, red, and blue segments, respectively. After replication, fusion, and centromere separation, the whole arm has been duplicated to form a palindrome, EDCBAABCDE, as in Figure 3.1b(c). This palindromic stretch of DNA is then torn apart. Unless it breaks in the center, between the two A segments, one daughter cell will have a deletion and the other will have an inverted duplication. In Figure 3.1b(d), one daughter cell gets CDE while the other gets BAABCDE. Thus, BFB cycles can be thought of as operations on a string of chromosomal segments. The only significant restriction this creates is that breaks must occur at segment boundaries, but this is not a loss of generality since the segment boundaries can be defined freely. Moreover, BFB breakpoints may be reused in subsequent BFB cycles [69], so it is likely useful to keep track of candidate breakpoints.

We are now ready to describe our model. Let Σ ($|\Sigma| = k$) be an alphabet where each symbol corresponds to a chromosomal segment, and the ordering of Σ corresponds to the ordering of the segments from telomere to centromere. Let x^t denote the string representing chromosomal segments after t BFB operations. Before any BFB operation, the string is just the initial segments of the chromosome in order. Therefore, x^0 consists of all k lexicographically ordered characters from Σ. Let x^{-1} be the reverse of string x, $\text{pref}(x)$ be a prefix of the string x, and $\text{suff}(x)$ be a suffix of the string x.

$$x^t = \begin{cases} \text{pref}(x^{(t-1)})^{-1}x^{(t-1)} & \text{if inverted duplication} \\ \text{suff}(x^{(t-1)}) & \text{if deletion} \end{cases} \tag{3.1}$$

Define a *BFB-schedule* as a specific sequence of BFB operations, that is, inverted prefix duplications and prefix deletions. Define x as a *BFB*(x^0)*-string* if it can be generated from x^0 through a series of BFB operations. For ease of notation, x^0 is implied, and we refer to x being a BFB-string. The following simple lemmas establish basic properties and ensure that we do not need to worry about deletions.

Lemma 1. *Let x be a BFB-string. If x^0 is a suffix of x, then x can be obtained from x^0 using only inverted prefix duplications.*

Proof. Suppose x is produced by a BFB schedule that includes a prefix deletion. The deletion must not delete any characters from the original string x^0 because if it did, there would be no way to subsequently generate those characters, and x^0 would not be a suffix of x.

If a prefix deletion removes some of the characters produced by an inverted prefix deletion, then the deletion's affect can be achieved by making the inverted prefix shorter. If the deletion removes all of the characters produced by an inverted prefix deletion, its affect can be achieved by omitting the inverted prefix deletion.

A prefix deletion will remove all or some of the characters produced by at least one previous prefix inverted duplication and all of the characters produced by any duplications after that inverted prefix duplication but before the prefix deletion. If the deletion is removed from the BFB schedule and the appropriate inverted duplications are removed or shortened, then the final string produced will remain the same. □

Lemma 2. Suffix Lemma: *Any suffix of a BFB-string is itself a BFB-string.*

Proof. A suffix of x can be made by a prefix deletion, which is a BFB operation. Since x can be achieved via BFB operations than so can a suffix of x. □

3.3 Algorithms for BFB

We begin with a simple problem: *Given string x of length n, determine if x is a BFB-string*. This can be solved in $O(n)$ time using the following algorithm:

```
Algorithm CheckBFB(x)
Input: String x (|x| = n) containing suffix x^0 (|x^0| = k)
Output: True if x is a BFB-string
1.  (* Find the longest even palindrome beginning at each character in x *)
2.  for (1 ≤ i < n − k)
3.          P[i] =max{ℓ | x[i...i+2ℓ] is palindromic})
4.  i = 1
5.  while (i < n − k)
6.          if (P[i] = 0) return false
7.          i = i+P[i]
8.  return true
```

Theorem 3. *Algorithm CheckBFB checks if x is a BFB-string in $O(n)$ time.*

Proof. (Sketch) A string is a BFB-string iff it can be formed by an inverted prefix duplication from another BFB-string or it is the original string, x^0. An inverted prefix duplication forms an even palindrome at the beginning of a string. CheckBFB finds such a palindrome and then, in effect, recurses on the string that ends at that palindrome's center. Lemma 2 guarantees that that string must be a BFB-string if x is a BFB-string. If a string does not begin with a palindrome, and it is not x^0, then it is not a BFB-string.

A linear time algorithm for finding maximum palindrome sizes is described in Appendix B. The remainder of CheckBFB visits each character at most once, so CheckBFB is in $O(n)$. □

For illustration, consider the palindrome array P_1 for the string $x_1 =$ BAABC-CBAAAABC. Starting with $i = 1$, we can advance i as $1 \rightarrow 3 \rightarrow 6 \rightarrow 10 \rightarrow 11 \rightarrow$ True. For $x_2 =$ BBCCCCBBBAABC, we advance i as $1 \rightarrow 5 \rightarrow 6 \rightarrow$ False.

i:	1	2	3	4	5	6	7	8	9	10	11	12	3
x_1	B	A	A	B	C	C	B	A	A	A	A	B	C
$P_1[i]$	2	4	3	2	1	4	3	2	1	1	0	0	0
x_2	B	B	C	C	C	C	B	B	B	A	A	B	C
$P_2[i]$	4	3	2	1	1	0	1	1	2	1	0	0	0

Thus, if we are given the full ordering of segments of a chromosome arm, we can determine if BFB could have produced that ordering. However, such complete information is often not available from current technologies. For example, an array CGH or sequencing experiment may only give the count of each segment, not the order. So, we would like to determine if a pattern of *copy counts* of chromosomal segments could have been produced by BFB. Formally, we define a sequence of positive integer copy counts from telomere to centromere along a chromosome arm as a *count-vector* ($\vec{n} = [n_1, n_2, ..., n_k]$). We say *$\vec{n}$ admits a BFB-schedule* if there is some BFB-string x whose character counts equal $\vec{n}$. For example, the count vector $[6,3,5]$ admits the BFB schedule

$$\underline{ABC} \rightarrow \underline{C}BAABC \rightarrow \underline{CCBAA}BC \rightarrow \underline{AA}BCCCCBAABC \rightarrow AAAABCCCCBAABC$$

In this case, x^4 has 6, 3, and 5 of characters A, B, and C. We now define our problem: *The BFB-count-vector Problem: Given a count vector $\vec{n}$, does $\vec{n}$ admit a BFB schedule?*

BFB-Pivot Algorithm: Each inverted prefix duplication reverses the order in which characters in the BFB-string appear. Consider the BFB schedule $\underline{ABC} \rightarrow \underline{CB}AABC \rightarrow BCCBAABC$. From left to right, the characters appear in proper lexicographical order, then reversed, then proper again. And each character is preceded by either itself or the next character higher or lower depending on whether the string is increasing or decreasing. With this knowledge, we can create a simple algorithm that adds single characters to the beginning of a candidate BFB-string until either a string with character counts satisfying $\vec{n}$ is found or one is shown not to exist.

```
Algorithm BFB-Pivot
Input: A count vector $\vec{n}$, string $s$
Output: True if $s$ can be extended via BFB operations to satisfy $\vec{n}$
1.  if not $CheckBFB(s)$
2.    then return false
3.  if $s$ satisfies $\vec{n}$
4.    then return true
5.  if count of $firstChar(s)$ in $s$ is even
6.    then $nextChar \leftarrow nextLetter(firstChar)$
7.    else $nextChar \leftarrow prevLetter(firstChar)$
8.  for $char \in \{nextChar, firstChar\}$
9.        if count of $char$ in s $\neq n[char]$
10.         then $s' = char + s$
11.             if BFB-Pivot($\vec{n}$, $s'$)
12.               then return true
13.         else continue
14. return false
```

Lemma 4. *BFB-Pivot$(\vec{n}, x)$ returns True if and only if x can be extended via BFB operations to a string with counts satisfying $\vec{n}$.*

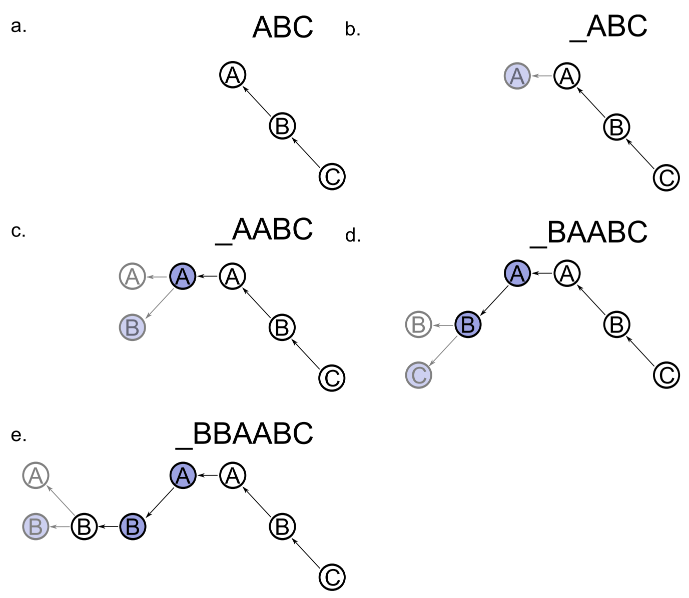


Figure 3.2: An illustration of BFB-Pivot searching for candidate BFB strings.

Proof. (Sketch) As we note above, segments of a BFB-string oscillate between "increasing" and "decreasing". So, when a character is prepended to an existing candidate BFB-string, it must either follow the existing trend, or it can be the beginning of a new inverted prefix duplication. In the latter case, another instance of the current first character of the string will be prepended.

BFB-Pivot attempts to prepend both eligible characters to the existing string until either an acceptable string is found or the current string fails because it is not a BFB-string or the count of a character exceeds the count in the count vector. By checking all such candidate BFB-strings, BFB-Pivot is guaranteed to find a BFB-string satisfying $\vec{n}$ if such a string exists. □

It is useful to consider a graphical representation of BFB-Pivot, shown in Figure 3.2. As nodes of the same character are added, the color of the added node oscillates. For each leftmost node, two possible edges are considered, a left edge ($\leftarrow$) and either an up ($\nwarrow$) or down ($\swarrow$) edge, depending on the color of the node. Therefore, the worst case complexity of BFB-Pivot is $O(2^n)$ where $n = \sum n_i$. Note that the input size of the problem is only $O(k \log n)$, so the output of a consistent BFB string as a certificate of correctness is already exponential in the input size.

BFB-trees We will now develop a second algorithm for solving the BFB-count-vector problem based on the equivalence between BFB-strings and trees with special symmetries. This new algorithm will have running times orders of magnitude lower than those of BFB-pivot in practice. First, we will prove some properties of BFB-strings.

Each BFB-string has a "lowest" character. For example, in the BFB-string CCCBBCCBAAAAAABC, that character is C. We will call this the *base character*. Base characters appear in pairs within a BFB-string, with all other characters appearing within these pairs. For example, the above BFB-string can be broken into CC, CBBC, and CBAAAAAABC. We will call these substrings bounded by base characters *return-blocks*. If the count of the base character of a BFB-string is even, then the entire BFB-string will be composed of return-blocks.

Lemma 5. *Every return-block is a palindrome.*

Proof. First, we will show that the first/rightmost return-block (RB) is a palindrome. Then, we will show that if all RBs to the right of a given RB are palindromes, then that RB is a palindrome as well, and we will have our result by induction.

BFB-strings are produced by inverted prefix duplications. As a result, they are composed of palindromes such that the left edge of one palindrome is the center of a subsequent palindrome. Consider the rightmost RB of a BFB-string. It is flanked by two base characters. The only palindrome these characters could be a part of is one centered between them, so the rightmost RB is a palindrome.

Now consider an RB where all RBs to the right are palindromes. Either a palindrome center lies within the RB or no centers lie within the RB. In the latter case, the RB is the reverse of a palindromic RB to the right, and hence a palindrome. In the former case, we have a situation similar to that of the rightmost RB: the left base character can only be part of a palindrome centered at the center of the RB. Thus, the RB is a palindrome. □

Lemma 6. *Every return-block is a BFB-string.*

Proof. The rightmost return-block is a BFB-string by the Suffix Lemma. Now suppose we have an RB such that all RBs to the right are BFB-strings. Either a palindrome center lies within the RB or no centers lie within the RB. In the latter case, the RB is the reverse of an RB to the right. By Lemma 5, that RB is a palindrome, and by assumption, it is a BFB-string. So, the RB must be a BFB-string as well.

If there are palindrome centers within the RB, then there is a palindrome with a left edge within the RB. The portion of that palindrome that lies within this RB is a suffix of an RB to the right, and hence is a BFB-string. Subsequent BFB operations that extend this suffix to the full RB still results in a BFB-string. □

Now consider a count-vector $\vec{n}$ that admits a BFB-schedule and the graph representation of the resulting BFB-string, as in Figure 3.2. Denote the layers of the graph corresponding to the characters as $\sigma_1, \sigma_2, \ldots, \sigma_k$. A count-vector $\vec{n}$ that admits a BFB-schedule and has a final count of 2 yields a return-block; we denote this an *RB-BFB-schedule*. A given count-vector may not end in a 2 and thus may not yield a BFB-string that is a single return-block. But, we can transform the BFB-string into a return-block through a single BFB operation:

Lemma 7. *The count-vector $\vec{n} = [n_1, n_2, \ldots, n_k]$ admits a BFB schedule iff $[2n_1, 2n_2, \ldots,$ $2n_k,\, n_{k+1} = 2]$ admits a BFB-schedule.*

Proof. If $[n_1, n_2, \ldots, n_k]$ admits a BFB-schedule, then adding a final inverted duplication of the entire string will achieve $[2n_1, 2n_2, \ldots, 2n_k, 2]$. If $[2n_1, 2n_2, \ldots, 2n_k, 2]$ admits a BFB-schedule, the corresponding BFB-string is composed of one return block and is thus a palindrome. Therefore, it has a suffix with counts $[n_1, n_2, \ldots, n_k]$, which by the Suffix Lemma is a BFB-string. □

As a result, we can focus on RB-BFB-schedules without loss of generality. Lemma 6 shows that BFB-strings have a recursive structure that allows us to represent them as trees (Figure 3.3b,c), Consider a BFB-string for a count-vector $\vec{n} = [n_1, n_2, \ldots, n_k]$, converted into an RB-BFB-string for $[2n_1, 2n_2, \ldots, 2n_k, n_{k+1} = 2]$ (Fig. 3.3b). Create a root r of the tree (with label corresponding to σ_{k+1}). The path through the BFB graph starts at σ_{k+1}, traverses other return-blocks at level k, and finally returns to σ_{k+1}. Each return-block at level k is the root of a subtree with r as its parent.

We use this idea to define rooted *labeled* trees. Each node is labeled so all nodes an identical distance from the root have the same label. For a node v with $2\ell + 1$ (odd) children, number the child nodes as $v_{-\ell}, \ldots v_0, \ldots, v_\ell$. For a node with 2ℓ children

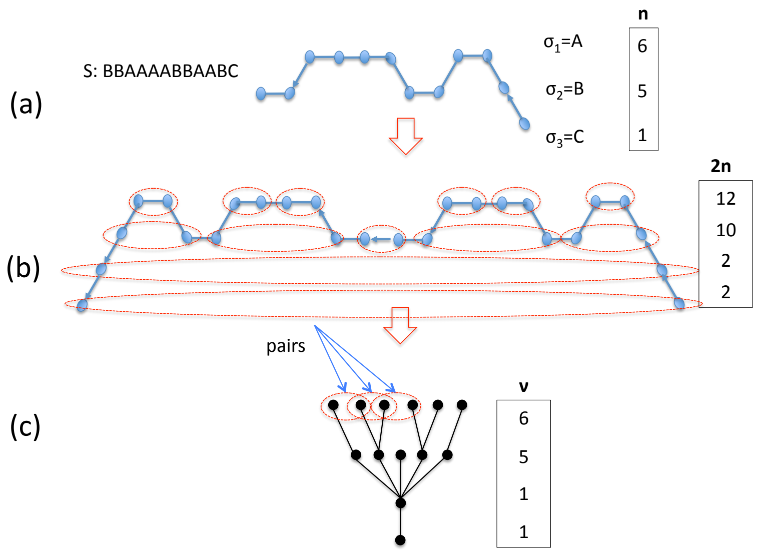


Figure 3.3: BFB-tree generated from an RB-BFB-schedule. (a) A graph for BBAAAABBAABC that supports $[6,5,1]$. (b) A single BFB operation begets an RB-BFB graph for $[12,10,2,2]$. Dotted ellipses denote the nodes of a BFB-tree. (c) A BFB-tree for the BFB-string, with 3 levels. The single node at level 3 has 5 children, ordered as $\{-2,-1,0,1,2\}$. Pairs are illustrated at levels 1, and 2.

the child nodes are labeled $v_{-\ell},\dots,v_{-1},v_1,\dots,v_\ell$; there is no v_0 node. $T(v)$ denotes the subtree rooted at v. Define a *labeled-traversal* of $T(r)$, as the string obtained by traversing the labels in an ordered, depth-first-search as given below:

Algorithm *LabelTraverse(T(r))*
Input: A labeled tree rooted at r
Output: The string given by a labeled traversal

```
1.  Let $\sigma = \text{Label}(r)$
2.  Let $S_1 = \varepsilon$
3.  for (each child $r_j$ ordered from least to max)
4.          $S_1 = S_1 \cdot \mathit{LabelTraverse}(T(r_j))$
5.  return $\sigma S_1 \sigma$
```

A labeled tree is *mirror-symmetric* if for all nodes v, LabelTraverse$(T(v))$= LabelTraverse$(T(v))^{-1}$. In other words, 'rotation' at v results in the same tree. The labeled traversal of a tree T visits each node exactly twice, outputting its label each time. Define a partial order on the nodes of T in which $u < v$ if the last appearance of u precedes the first appearance of v in the labeled traversal. For any $u < v$, let $T_<(u,v)$ denote the subtree of T containing the LCA of u,v and all nodes w s.t. $u < w < v$. We call (u,v), with $u < v$, a *label-pair* if u and v have the same label σ, and no node in $T_<(u,v)$ has label σ. A labeled tree is *pair-symmetric* if for all label-pairs (u,v), $T_<(u,v)$ is mirror-symmetric. Finally, we say a labeled tree has *long-ends* if starting from the root, and following the least numbered node at each step, we can reach each layer $k,k-1,\dots,2,1$.

Definition 1. *A* BFB-tree *is a labeled tree with long-ends, mirror-symmetry, and pair-symmetry.*

Theorem 8. *Let $T(r)$ be a BFB-tree rooted at r. Then* LabeledTraversal$(T(r))$ *is an RB-BFB-string.*

Proof. Recall that a BFB-string is a string composed of overlapping palindromes such that the left edge of one palindrome is the center of the subsequent palindrome. Because $T(r)$ is a BFB-tree, it is mirror-symmetric. Thus, the label traversal beginning from r

will yield a string composed of nested palindromes. And, at any level of the tree, the label traversal will yield a set of concatenated palindromic return-blocks.

Because $T(r)$ has long-ends, we know that the label traversal of each return-block does not contain any characters lexicographically higher than its initial ordered substring. For example, the return-block DCBBCCBBCD begins with BCD and hence cannot contain any A's. So, if there are two consecutive return-blocks such that the highest character in the return-block on the left is not higher than the highest character in the return-block on the right, there is a palindrome centered between the two return-blocks that extends to another palindrome center on the left. For example, if we have DCBBC-CBBCDDCBAABCD, we have two return-blocks: DCBBCCBBCD and DCBAABCD. And, between them, there is the palindrome BCDDCB whose left edge is the center of the palindrome CBBC. So, we are able to find a set of appropriately overlapping palindromes.

On the other hand, if there are two consecutive return blocks such that the highest character in the return-block on the left *is* higher than the highest character in the return-block on the right, pair-symmetry guarantees that there is still a palindrome that extends left to another palindrome center. For example, the string CBAABCCBBCCBAABC is composed of three return-blocks: CBAABC, CBBC, and CBAABC. Pair-symmetry ensures that there is a palindrome that reaches to the center of the leftmost return-block, in this case ABCCBBCCBA.

Thus, the structure of $T(r)$ guarantees that its label traversal is composed of overlapping palindromes and is thus a BFB-string.

□

Theorem 9. *The tree T_S derived from an RB-BFB-string S is a BFB-tree.*

Proof. By Lemma 5, we have that each return-block is palindromic. This ensures that the label traversal of the tree rooted at any node is palindromic, and thus the T_S has mirror-symmetry. By Lemma 6, we have that each return-block is a BFB-string. By definition, a BFB-string begins with all characters that appear in the string in lexicographical order. Thus, from any node v in T_S, we can follow edges to the least- numbered child to reach the deepest node in $T_S(v)$. Hence, T_S has long-ends. Finally, if there are whole return-blocks between two consecutive instances of the same character in S, then the concatenation of those return-blocks must be palindromic. Otherwise, no palindrome could include both instances of the character. Thus, T_S has pair-symmetry. $\square$

The BFB-Tree Algorithm: Given a count-vector $\vec{n} = [n_1, n_2, \ldots, n_k]$ ($\sum_i n_i = n$), the BFB-Tree algorithm builds a BFB-tree on $n+1$ nodes, with the count of nodes in each layer given by $\vec{v} = [n_1, n_2, \ldots, n_k, 1]$. We start with the single node BFB-tree T at level $k+1$, and extend it layer by layer. In each step j, $1 \leq j \leq k$, we assign n_j children to the leaves of the current tree T, maintaining BFB-properties.

Denote the children of node v by the set $C_v = \{v_{-\ell}, \ldots, v_\ell\}$. Mirror-symmetry ensures that for each $0 \leq i \leq \ell$, the subtrees $T(v_{-i})$ and $T(v_i)$ are identical. Thus, it can be said the subtree $T(v_{-i})$ is *dependent* on the subtree $T(v_i)$. We maintain this information by defining *multiplicity,* $I(v)$, for each node as follows: $I(r) = 1$ for the root node r. For nodes v with children $v_{-\ell}, \ldots, v_\ell$, set $I(v_j) = 2I(v)$ for $\ell \geq j > 0, I(v_0) = I(v)$, $I(v_j) = 0$ otherwise. That is, for child nodes v_j with a corresponding dependent child v_{-j} we assign a multiplicity of double the parent's multiplicity. For nodes with an odd number of children, there will be a child v_0 without a corresponding dependant child node. This node is assigned the same multiplicity as the parent node. Finally, for each dependent node pair, one is assigned a multiplicity of zero since, by mirror-symmetry, it is completely defined by its dependent whose multiplicity has doubled. Valid assignments at level j must then satisfy the diophantine equations $\sum_v I_v = n_{j+1}$, and $\sum_v I(v)|C_v| = n_j$.

Algorithm *BFB-Tree*(T, j, I)

```
1.  **if** $j = 0$, return CheckBFB(LabeledTraversal(T))
2.  **for** each assignment $C_v$ s.t. $\sum_v I(v)|C_v| = n_j$
3.          Extend $T$ according to the assignment
4.          Adjust $I$
5.          **if** (Construct_BFB_tree($T, j-1, I$)) **return true**
6.  **return false**
```

Note that once a node is dependent ($I(v) = 0$), its descendants remain dependent. Further, the multiplicity of the node is a power of 2, and is doubled each time an independent node is a non-central child of its parent. As the multiplicities increase, the number of valid assignments decreases quickly, improving the running time in practice. Further analysis of the algorithm is presented in Appendix .

Rules for BFB: In addition to the two algorithms for solving the BFB count-vector problem presented above, it is also possible that some combinatorial rules completely define the set of count-vectors that admit a BFB schedule. We present seven conditions below that can guarantee that a BFB-schedule exists or does not exist for a subset of possible count-vectors.

Consider a count-vector $\vec{n} = [n_1, n_2, \ldots, n_k]$. We use *BFB($\vec{n}$)* to denote that $\vec{n}$ admits a BFB schedule.

Lemma 10. *[Subsequence Rule] Let $\vec{n}'$ be a subsequence of $\vec{n}$. BFB$(\vec{n}) \Rightarrow$ BFB$(\vec{n}')$.*

For example, if (6,4,6,8) admits a BFB schedule, then so must (6,4,6), (6,6,8), (6,4,8), (4,6,8), etc.

Proof. As discussed above, a BFB string is composed of palindromes that overlap so that the end of one palindrome is the center of the next palindrome. If all instances of a particular character are removed, the palindromes will still exist and have this property. So, the string will still be a BFB string. And, the character counts associated with the string will be a subsequence of the counts of the original string. $\square$

Lemma 11. *[Rule of One] If $\exists\, i < j \leq k$ such that $n_i = 1$, $n_j > 1$, then $\neg$BFB$(\vec{n})$.*

Proof. In order for the j^{th} character to achieve a count greater than one, it must be part of a prefix inversion/duplication. If the j^{th} character is in the prefix, then so is the i^{th}. But, the i^{th} character has a count of one, so it can never be part of a prefix inversion/duplication. $\square$

Lemma 12. *[Odd-Even Rule] If $\exists\, i < j \leq k$ such that n_i is odd and n_j is even, then $\neg$BFB$(\vec{n})$.*

Proof. We will consider the case where $k = 2$ and n_1 is odd and n_2 is even. So, we start with AB and need to get an odd count of As and an even count of B's.

We will show that, in a string yielded by any BFB schedule, the number of A's to the right of any B is even. First consider the trivial case where the B in question is the rightmost B. Then the number of A's to the right of that B is zero, which is even.

Now, suppose there is only one run A's to the right of the B in question, B_1. Then the string is

$$B_1(B)_m(A)_nB_2$$

where $m >= 0$ and $(A)_n$ means a string of n A's.

In this case, B_1 can only have been generated after a copying of B_2. When B_2 was copied, it doubled the count of A's before B_2. So, n must be even.

Finally consider the case when there are arbitrarily many runs of A's after the given B, and all but the leftmost run contain an even number of A's

$$\dots B_1(B)_n(A)_k(B)_n(A)_{2i}(B)_\ell \dots (A)_{2j}B_2$$

In this case B_1 could only have been generated after a copying of a run of B's to the right of $(A)_k$. This copying doubled the previous number of A's, so k is even. Since k is even and the counts of all other runs of A's are even, then the total count of A's to the right of B_1 is even. The result that the number of A's to the right of any B is even then follows by induction.

Similar reasoning can establish that the count of B's to the right of any A is odd. When a BFB schedule begins, A can be duplicated arbitrarily many times before B is duplicated for the first time, yielding

$$B(A)_mB$$

In this case, the count of B's to the right of any A is 1, which is odd. Now, consider the case where there are only two runs of B's after a given A. One of those runs must be the original B, so we have

$$\dots A_1(A)_k(B)_n(A)_mB$$

Using the reasoning above, n must be even because A_1 could only have been generated after a copying of an A from $(A)_m$, which would have doubled the B's leading to the run $(B)_n$.

In the case where there are arbitrarily many runs of B's after an A, and all of the runs have an even number of B's except the leftmost run and the rightmost run of length 1. Then, the count of B's to the right of every A is odd.

Now, return to the question of whether a BFB schedule can create a final string with an odd number of A's and and even number of B's. The final string ends in either A or B. If it ends in A, then the count of B's is odd because the count of B's to the right of the last A must be odd. But, we want the count of B's to be even, so the string cannot end in A. If the string ends in B, then the total count of A's must be even, but we want the count of A's to be odd. So, there is no string resulting from a BFB schedule where the count of A is odd and the count of B is even.

By using the Subsequence Rule, we can extend this result to conclude that no odd count can precede an even count in a count-vector achievable by BFB. □

Lemma 13. *[Rule of Four] Suppose, all counts in $\vec{n}$ are even. Let i be the index of the first count that is not divisible by four, that is $4 \nmid n_i$ and $\forall\ j < i\ 4|j$. Let f_ℓ be the number of times that divisibility by four changes in $n_i, \ldots, n_\ell$. If $f_\ell > \frac{n_\ell}{2}$ then $\neg BFB(\vec{n})$.*

Proof. Suppose that $f_\ell > \frac{n_\ell}{2}$. Consider only counts that come before n_ℓ so that the character corresponding to n_ℓ is the base character. Then, the corresponding BFB-string will be composed of $\frac{n_\ell}{2}$ return-blocks. By Lemmas 5 and 6, each return-block is a palindrome and a BFB-string. By the Suffix Lemma, the right half of each return-block is also a BFB-string. The Odd-Even Rule tells us that no odd count precedes an even count in each half-return-block. So, no count not divisible by four precedes a count divisible by four within a return-block. Thus, after each return-block, the number of times that divisibility by four changes in the remaining counts can be decremented by at most one. □

Lemma 14. *[Two Reduction] If $n_i = 2$ for some i. Then,*
$BFB(\vec{n}) \Leftrightarrow BFB([\frac{1}{2}n_1, \ldots, \frac{1}{2}n_{i-1}]) \bigwedge BFB([n_k - 1, n_{k-1} - 1, \ldots, n_i - 1])$

Proof. Consider a set of counts $a, b, 2, d, e$ that starts with the string ABCDE. At some point, the C must be part of a prefix inversion/duplication so that it can achieve a count of 2. After that, no C can be part of any subsequent prefix inversion/duplication, as that would increase its count above 2. At that point, all A's and B's will be after a C, so no A or B can be part of a subsequent prefix inversion/duplication either. And, the duplication of the C would have also duplicated all existing A's and B's. So, a BFB schedule yielding $\frac{a}{2}, \frac{b}{2}$ must be part of the overall BFB schedule yielding $a, b, 2, d, e$.

Now, if $d > 1$ or $e > 1$, then D and E must also be duplicated the first time C is duplicated. If they were not, then the string after the C duplication would be C[AB]*CDE, and all D's and E's would be after a C and therefore ineligible to be part of any prefix inversion/duplication. So, the string must be EDC[AB]*CDE, and subsequent prefix inversion/duplications will not extend past the final ED. The subsequent BFB schedule must achieve counts of $e-1$ and $d-1$ for E and D, respectively. $\square$

Lemma 15. *[Four Reduction] If $\exists\ i < j < k$ s.t. $n_i = 4$, $n_k = 4$, and $n_j > 4$. Then $BFB(\vec{n}) \Rightarrow BFB(\frac{n_1}{2}, \frac{n_2}{2}, \ldots, \frac{n_i}{2})$.*

Lemma 16. *[Odd Reduction] Suppose all counts in $\vec{n}$ are odd. Then, $BFB(\vec{n}) \Rightarrow BFB(reverse(\vec{n} - \vec{1}))$.*

We also have a sufficient condition for BFB($\vec{n}$).

Lemma 17. *[Count Threshold Rule] $BFB(\vec{n})$ if for all i n_i is even, and $n_i > 2(i-1)$.*

Once the rules have been applied, we apply BFB-Pivot or BFB-Tree. Note that some of these rules are *reductions*, that is, they reduce an instance of the count-vector problem to simpler instances. These can be applied multiple times. More detail is available in Appendix B.

3.4 Results

Performance on realistic data-sets Using practical sized examples, we investigated the performance of the 3 approaches to checking BFB: rules, BFB-Pivot, and BFB-Tree. We applied the BFB rules to all $\sum_{i=1}^{5} 20^i = 3,368,420$ count-vectors with $k \leq 5$ and each $n_i \leq 20$. Remarkably, the rules were able to resolve $3,034,440$, or 90%, of the count-vectors. As BFB-Pivot and BFB-Tree are both exponential time procedures, this check helped speed up the entire study.

Nearly all of the count-vectors that the rules could not resolve admitted a BFB schedule (Table 3.1), so BFB-Pivot and BFB-Tree could usually halt once an acceptable BFB string was found rather than exhausting all possible paths or trees.

Table 3.1: Method and result for the count-vectors used to analyze algorithm speed.

	Rules	Tree/Pivot	Total	
Admits BFB	170,576	333,840	504,416	(15.0%)
Not BFB	2,863,864	140	2,864,004	(85.0%)
Total	3,034,440	333,980	3,368,420	
Fraction	(90.1%)	(9.9%)		

We ran BFB-Pivot and BFB-Tree on each of the 333,980 count-vectors that could not be resolved by rules. The running times plotted against n for each algorithm is shown in Figure 3.4. As expected, both algorithms' worst-case running times grew exponentially with n. However, the worst case running times for BFB-Tree were orders of magnitude lower than for BFB-Pivot. For example, the longest running count-vector for BFB-Tree was [20,3,19,19,19] which took 10 seconds to complete. The longest running count-vector for BFB-Pivot was [20,18,20,20,6] which took 23,790 seconds to complete.

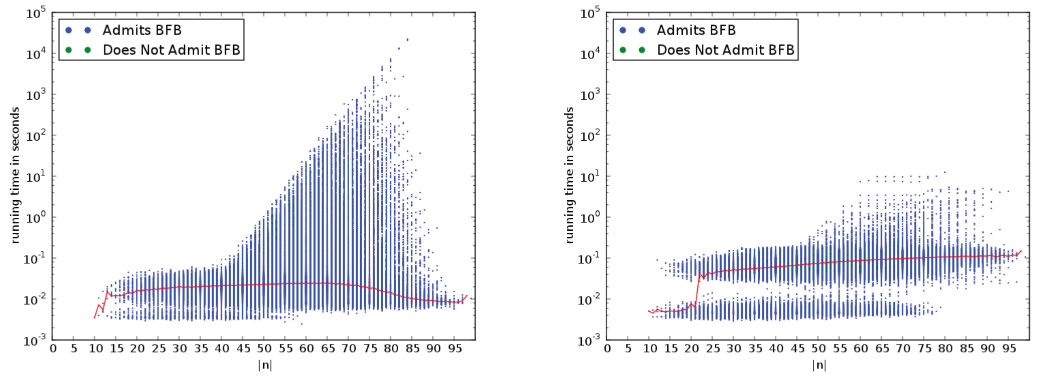


Figure 3.4: Pivot (left) and tree (right) algorithm running time. Each point refers to the size and running time of a specific example. The line represents median running time. Note that since the rules excluded nearly all count-vectors that did not admit a BFB schedule, almost all points are blue.

Consequences For Experimental Interpretation Consider experimental data, such as array CGH or read mapping depth of coverage, that reveal the copy counts of segments of a chromosome arm, or in the terminology of this chapter, a count-vector. If this count-vector admits a BFB schedule, one might infer from this observation that BFB occurred.

On its face, this is a reasonable conclusion. Only 15% of the count-vectors with $k \leq 5$ admit a BFB schedule. If we expand that analysis to the 64,000,000 count-vectors with $k = 6$ and $n_i \leq 20$, only 7.3% admit a BFB schedule. So, it is plausible that observing a count-vector that admits a BFB schedule is much more likely if BFB did in fact occur. However, using our model of BFB, we show that this inference is usually incorrect.

Note first that experimentally-derived count-vectors are often imprecise. Experiment typically provides a small range of values for each element in the count-vector rather than a single definite value. This is a result of the imprecision of the experimental method as well as potentially high levels of structural variation or aneuploidy in the genome being studied. Therefore, the observation made about the data may not be that a particular count-vector admits a BFB schedule but that there is a count-vector that admits a BFB schedule "nearby", that is, within experimental precision of the observed count-vector.

Further, the uncertainty in a count tends to increase with the magnitude of the count. It is easier to distinguish between copy counts 1 vs. 2 than between 32 vs. 33. For simplicity, we assume that the relationship between uncertainty and magnitude is linear and use the Canberra distance [18] to compare count-vectors:

$$d(x,y) = \sum_i \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{3.2}$$

For each of the 64,000,000 count-vectors with 6 segments, we searched for the nearest count-vector that admits a BFB schedule and recorded the distance to that count-vector. For the 7.2% of count-vectors that admit a BFB schedule, the distance was, of course, zero. The distribution of distances is shown in Figure 3.5.

The results are striking. Consider the count-vector $[14,7,18,16,9,12]$, which does not admit a BFB-schedule. The nearest count-vector that does is [14,7,19,17,9,13],

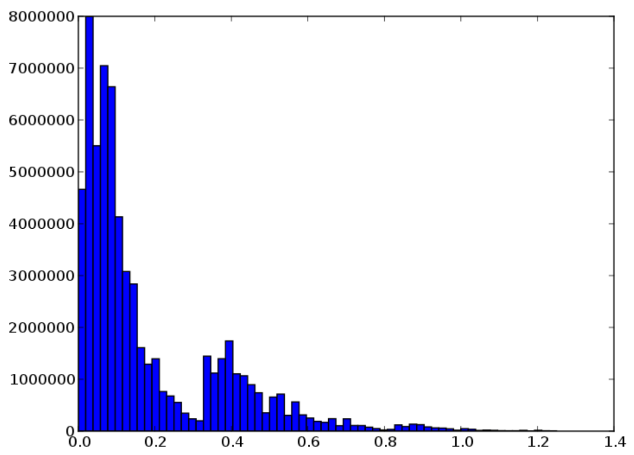


Figure 3.5: Distribution of distances to nearest count-vector admitting a BFB schedule.

Table 3.2: Percentage of count-vectors at least as close to a count-vector admitting a BFB schedule as the shown count-vector pair.

Distance (%ile)	Count-Vector	Nearest Admitting BFB
.097 (50)	[14,7,18,16,9,12]	[14,7,19,17,9,13]
.129 (60)	[7,13,6,4,19,12]	[8,14,6,4,20,12]
.192 (70)	[9,7,7,8,2,14]	[8,8,8,8,2,14]
.362 (80)	[16,17,14,18,1,19]	[16,18,14,18,2,19]
.458 (90)	[20,1,7,3,10,6]	[20,2,7,3,11,7]
.566 (95)	[15,8,8,1,15,2]	[16,8,8,2,15,3]
.889 (99)	[9,1,5,9,1,15]	[10,2,5,9,3,15]

at a distance of .097. Half of all count-vectors tested were at least this close to a count-vector admitting a BFB-schedule. So, if the precision of the experimental method employed is such that it can not reliably discern between a copy count of 12 and a copy count of 13 or a copy count of 18 and a copy count of 19, then half of the count-vectors we examined would appear to admit a BFB schedule. Similarly, if the method can not reliably discern a copy count of 9 and and 8 or 7 and 8, then 70% of count-vectors will appear to admit a BFB schedule. Figure 3.2 has distances and and example vector pairs for additional percentiles.

Thus, even with small amounts experimental uncertainty, a majority of count-vectors admit a BFB schedule, so mechanisms other than BFB are likely to produce count-vectors that look like they were created by BFB. Therefore, finding a count-vector consistent with BFB should only slightly increase one's belief that BFB occurred.

On the other hand, observing a count-vector that is *distant* from *any* BFB admitting count-vector, provides strong evidence that BFB was *not* the cause of the amplification. The likelihood of observing a count-vector that is distant from a count-vector that admits a BFB schedule is low in any case, and is surely even lower if the chromosome arm underwent BFB.

3.5 Discussion

We present perhaps the first formalization of the BFB mechanism. Our main result is that BFB can result in a surprisingly broad range of amplification patterns along a chromosome arm. Indeed, for most contiguous patterns of amplification, it is possible to find a BFB schedule that yields either the given pattern or one that is very similar. As a result, one must be cautious when interpreting copy count data as evidence for BFB. The presence of amplification at all, or the presence of a terminal deletion from a chromosome arm can both suggest the occurrence of BFB, though they may not distinguish well between BFB and other amplification hypotheses. However, unless the counts are known precisely, a specific pattern of copy counts along the chromosome arm cannot offer compelling support for BFB.

This study suggests several avenues for future work. There are other types of evidence that can be deployed to argue that BFB has occurred. FISH can reveal, to some extent, the arrangement of segments along a chromosome arm. Next-generation sequencing can reveal the copy counts of breakpoints between rearranged chromosomal segments and may allow for a fuller characterization of a chromosome arm. Methods similar to those presented in this chapter may prove useful for evaluating and interpreting these different types of evidence. Modeling may also be helpful for evaluating proposed refinements to models of BFB.

Finally, we have outlined two algorithms for determining whether a count-vector admits a BFB schedule, but neither is polynomial in reasonable measures of the input size. This problem, along with related problems, is interesting as computational problems *per se*. We hope in the future to have either faster solutions to these problems or proofs of hardness.

3.6 Acknowledgements

This research was supported in part by grants from the NIH (5RO1-HG004962) and the NSF (CCF-1115206).

Chapter 3 (with Appendix B) was published in the *Journal of Computational Biology*, Volume 19, Issue 6, pages 662–678, 2012, M. Kinsella and V. Bafna, “Combinatorics of the Breakage-Fusion-Bridge Mechanism” and was presented at the 16th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2012). The dissertation author was the primary investigator and author of this paper.

Chapter 4

An algorithmic approach for breakage-fusion-bridge detection in tumor genomes

4.1 Introduction

Genomic instability allows cells to acquire the functional capabilities needed to become cancerous [26], so understanding the origin and operation of genomic instability is crucial to finding effective treatments for cancer. Numerous mechanisms of genomic instability have been proposed [28], including the faulty repair of double-stranded DNA breaks by recombination or end-joining and polymerase hopping caused by replication fork collapse [12]. These mechanisms are generally not directly observable, so their elucidation requires the deciphering of often subtle clues after genomic instability has ceased.

In contrast, the breakage-fusion-bridge (BFB) mechanism creates gross chromosomal abnormalities that can be seen in progress using methods that have been available for decades [53]. BFB begins when a chromosome loses a telomere (Figs. 4.1a, 4.1b). Then during replication, the two sister chromatids of the telomere-lacking chromosome fuse together (Figs. 4.1c, 4.1d). During anaphase, as the centromeres of the chromosome migrate to opposite ends of the cell (Fig. 4.1e), the fused chromatids are torn apart

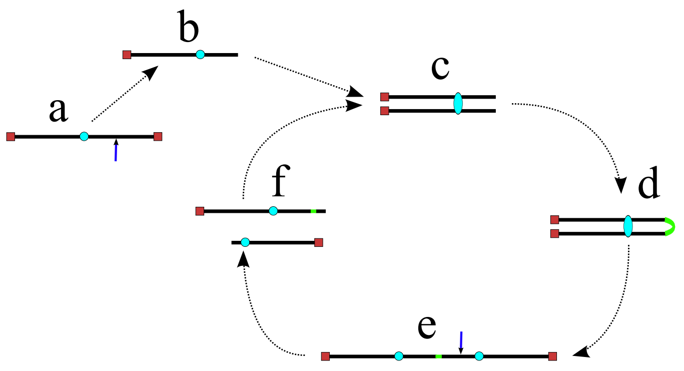


Figure 4.1: A schematic BFB process.

(Fig. 4.1f). Each daughter cell receives a chromosome missing a telomere, and the cycle can begin again. As this process repeats, it can lead to the rapid accumulation of amplifications and rearrangements that facilitates the transition to malignancy [17].

This process produces several plainly identifiable cytogenetic signatures such as anaphase bridges and dicentric chromosomes. However, as cancer genomics has shifted to high-throughput techniques, the signatures of BFB have become less clear. Methods like microarrays and sequencing do not allow for direct observation of BFB; instead BFB is now similar to other mechanisms of instability in that it must be inferred by finding its footprint in complex data.

Multiple groups have begun to address the problem of finding evidence for BFB in high-throughput data. For example, Bignell *et al.* found a pattern of inversions and exponentially increasing copy numbers "[bearing] all the architectural hallmarks at the sequence level" of BFB [7]. Kitada and Yamasaki found a pattern of copy counts and segment organization consistent with a particular set of BFB cycles [37]. Hillmer *et al.* used paired-end sequencing to find patterns of inversions and amplification explainable by BFB [30].

The procedures of these investigators, among others [46, 29, 68], share an element in common: they determine whether a particular observation is consistent with or could be explained by BFB. While this is helpful, it does not on its own allow one to infer whether or not BFB occurred. Indeed, in a previous work [36] we examined short patterns of copy number increases consisting of five or six chromosome segments. We found that most such patterns, whether produced by BFB or not, were consistent or nearly consistent with BFB. Thus, finding that such a pattern was consistent with BFB would only be weak evidence that it had been produced by BFB. This finding highlights the need for a rigorous and systematic approach to the interpretation of modern data for BFB in order to avoid being misled by the complexity of cancer genomes and the BFB mechanism itself.

Here we present a framework for interpreting high-throughput data for signatures of BFB. We incorporate observations of breakpoints as well as copy numbers to create a scoring scheme for chromosomes. Through simulations, we find appropriate threshold scores for labeling a chromosome as having undergone BFB based on varying models of cancer genome evolution and tolerances for error. This framework complements the work of previous groups by not only finding breakpoint and copy number patterns consistent with BFB but also showing under what assumptions they are more likely to be observed if BFB occurred than if it did not.

The key technical contribution that underlies our scoring scheme is a new, fast algorithm for determining if a given pattern of copy counts is consistent with BFB. This algorithm is related to a previously described algorithm [36] in that it takes advantage of a distinctive feature of BFB: when fused chromatids are torn apart, they may not tear at the site of fusion. This yields chromosomes with either a terminal deletion or a terminal inverted duplication. When a chromosome undergoes this process repeatedly, it results in particular patterns of copy number increases. The running time of the earlier algorithm grew exponentially with the amount of amplification and the number of segments in a copy number pattern. This greatly narrowed the scope of copy number patterns that could be investigated. This was particularly limiting because it appeared that copy number patterns with more segments would be more useful for identifying BFB, but these patterns could not be evaluated in a reasonable amount of time with the previous method. The new algorithm presented here is linear time and therefore allows complex copy number patterns to be checked in a trivial amount of time.

We begin by describing the kinds of high-throughput data that can provide evidence for BFB. We then proceed to lay out some formalizations needed to precisely describe scoring methods of samples based on BFB evidence implied from such data. Next, we define related computational problems, followed by an outline of algorithms for these problems. In the results section, we detail the simulations we used to measure the performance of our scoring system for BFB. Based on simulation parameters, we find false and true positive rates for different BFB signatures. We apply our methods to two datasets. The first is copy number data from 746 cancer cell lines [6]. We find three chromosomes that have long copy number patterns consistent with BFB, but the false positive rates from our simulations suggest that these may be false discoveries. We also examine paired-end sequencing data from pancreatic cancers [9]. We find two chromosomes that likely have undergone BFB, one that was identified by the original publishers of the data and one novel finding.

4.2 High-throughput evidence for BFB

We consider two experimental sources for evidence for BFB: microarrays and sequencing. Microarrays allow for the estimation of the copy number of segments of a chromosome by measuring probe intensities [13]. Sequencing also yields copy number estimates by measuring depth of sequence coverage [15]. In addition, if the sequencing uses paired-end reads and is performed on the whole genome rather than, say, the exome, it can reveal genomic breakpoints where different portions of the genome are unexpectedly adjacent. This is generally the extent of evidence available from either technique. Sequencing does not allow for a full reconstruction of a rearranged chromosome, as the repetitive nature of the genome leads to multiple alternative assemblies. Neither method can resolve segment copy numbers by orientation, so copy numbers from both forward and reversed chromosome segments are summed. Nevertheless, BFB should leave its signature in both breakpoints and copy counts, and we examine each in turn.

4.2.1 Breakpoints

During BFB, the telomere-lacking sister chromatids are fused together. This causes the ends of the sister chromatids to become adjacent but in opposite orientations (see Fig. 4.1d). This adjacency is unlikely to be disrupted by subsequent BFB cycles and will remain in the final sequence as two duplicated segments arranged head-to-head. If the chromosome is paired-end sequenced, the rearrangement will appear as two ends that map very near each other but in opposite orientations. This type of rearrangement has been termed a “fold-back inversion” [9], and regions of a chromosome rearranged by BFB should have an enrichment of these fold-back inversions. Reliable indications for fold-back inversions may or may not be available, depending on the type of experiment and its intensity.

4.2.2 Copy counts

Each BFB cycle duplicates some telomeric portion of the chromosome undergoing BFB. These repeated duplications should lead to certain characteristic copy number patterns, which are the signature of BFB in copy number data. We would like to evaluate copy numbers observed from microarrays or sequencing and determine if the copy numbers contain the footprint of BFB. Previous groups have searched for such a footprint by manually inspecting copy number data and searching for a set of BFB cycles that could produce the observed copy numbers [7, 37]. This approach is challenging and labor intensive, but developing a more general approach turns out to be rather difficult. A key technical contribution of this chapter is the development of efficient algorithms to evaluate copy counts for consistency with BFB.

4.2.3 Formalizing BFB

Creating an efficient method for evaluating copy numbers requires some formalization, so we begin with some definitions and basic results.

We represent a chromosome as a string ABC..., where each letter corresponds to a contiguous segment of the chromosome. For example, the string ABCD would symbolize a chromosome arm composed of four segments, where A is the segment nearest the centromere. More generally, we use σ_l for the l-th segment in a chromosome. So, ABCD could be written $\sigma_1\sigma_2\sigma_3\sigma_4$. A bar notation, $\bar{\sigma}$, is used to signify that a segment is reversed. Greek letters $\alpha, \beta, \gamma, \rho$ denote concatenations of chromosomal segments, and a bar will again mean that the concatenation is reversed. For example if $\alpha = \sigma_1\sigma_3\bar{\sigma}_2$, $\bar{\alpha} = \sigma_2\bar{\sigma}_3\bar{\sigma}_1$. An empty string is denoted by ε.

Consider the following BFB cycle on a chromosome X $\bullet$ ABCD, where $\bullet$ represents the centromere, X is one chromosomal arm, and ABCD is the 4-segmented other chromosomal arm which has lost a telomere. The cycle starts with the duplication of the chromosome into two sister chromatids and their fusion at the ends of the 'D' segments, generating the dicentric chromosome $\mathrm{X} \bullet \mathrm{ABCD\bar{D}\bar{C}\bar{B}\bar{A}} \bullet \bar{\mathrm{X}}$. During anaphase, the two centromeres migrate to opposite poles of the cell and a breakage of the dicentric chromosome occurs between the centromeres, say between $\bar{\mathrm{D}}$ and $\bar{\mathrm{C}}$, providing one daughter cell with a chromosome with an inverted suffix, $\mathrm{X} \bullet \mathrm{ABCD}\bar{\mathbf{D}}$, and another daughter cell with the trimmed chromosome $\mathrm{X} \bullet \mathrm{ABC}$ (chromosomes $\mathrm{\bar{C}\bar{B}\bar{A}} \bullet \bar{\mathrm{X}}$ and $\mathrm{X} \bullet \mathrm{ABC}$ are equivalent). The now amplified segment D in the first daughter cell may confer some proliferative advantage, causing its descendants to increase in frequency. The daughter cells also lack a telomere on one chromosome arm and therefore may undergo additional BFB cycles. One possible subsequent cycle could, for example, cause an inverted duplication of the suffix $\mathrm{CD\bar{D}}$, yielding the chromosome $\mathrm{X} \bullet \mathrm{ABCD\bar{D}}\mathbf{D}\bar{\mathbf{D}}\bar{\mathbf{C}}$. As these BFB cycles continue, the count of segments on the modified chromosome arm can increase significantly.

The notation $\alpha \xrightarrow{\mathrm{BFB}} \beta$ will be used for indicating that the string β can be obtained by applying 0 or more BFB cycles over the string α, as formally described in Definition 2.

Definition 2.*For two strings α, β, say that $\alpha \xrightarrow{BFB} \beta$ if $\beta = \alpha$, or $\alpha = \rho\gamma$ for some strings ρ, γ such that $\gamma \neq \varepsilon$, and $\rho\gamma\bar{\gamma} \xrightarrow{BFB} \beta$.*

We say that β is an *l-BFB string* if for some consecutive chromosomal region $\alpha = \sigma_l\sigma_{l+1}\ldots$ starting at the l-th segment σ_l, $\alpha \xrightarrow{\mathrm{BFB}} \beta$. Say that β is a *BFB string* if it is an l-BFB string for some l. As examples, $\mathrm{CDE} = \sigma_3\sigma_4\sigma_5$ is a 3-BFB string, and so are $\mathrm{CDE}\bar{\mathbf{E}}$ and $\mathrm{CDE\bar{E}}\mathbf{E}\bar{\mathbf{E}}\bar{\mathbf{D}}$. The empty string ε is considered an l-BFB string for every integer $l > 0$.

Denote by $\vec{n}(\alpha) = [n_1, n_2, \ldots, n_k]$ the *count vector* of α, where α represents a modified chromosomal arm $\sigma_1\sigma_2\ldots\sigma_k$ with k segments, and n_l is the count of occurrences (or *copy number*) of σ_l and $\bar{\sigma}_l$ in α. For example, for $\alpha = \mathrm{BCD\bar{D}\bar{C}C}$, $\vec{n}(\alpha) = [0, 1, 3, 2]$. Say that a vector $\vec{n}$ is a *BFB count vector* if there exists some 1-BFB string α such that $\vec{n} = \vec{n}(\alpha)$.

4.2.4 Handling experimental imprecision

Experimental methods do not provide the precise and accurate copy number of a given chromosome segment. Instead, some measurement error is expected. Moreover, in a cancer genome, it is plausible that a region undergoing BFB may also be rearranged by other mechanisms. So when we evaluate a count vector for consistency with BFB, we must also consider whether the count vector is "nearly" consistent with BFB.

For this, we define a distance measure δ between count vectors, where $\delta(\vec{n},\vec{n}')$ reflects a penalty for assuming that the real copy counts are $\vec{n}'$ while the measured counts are $\vec{n}$. We have implemented such a distance measure based on the *Poisson likelihood* of the observation, as follows: Let $\Pr(n|n') = \frac{n'^{n}e^{-n'}}{n!}$ be the Poisson probability of measuring a copy number n, given that the segment's true copy number is n'. Assuming measurement errors are independent, the probability for measuring a count vector $\vec{n} = [n_1, n_2, \dots, n_k]$, where the true counts are $\vec{n}' = [n'_1, n'_2, \dots, n'_k]$ is given by $\Pr(\vec{n}|\vec{n}') = \prod\limits_{1 \le i \le k} \Pr(n_k|n'_k)$. Define the *distance* of $\vec{n}$ from $\vec{n}'$ by

$$\delta(\vec{n},\vec{n}') = 1 - \frac{\Pr(\vec{n}|\vec{n}')}{\Pr(\vec{n}'|\vec{n}')}$$

For every pair of count vectors $\vec{n}$ and $\vec{n}'$ of the same length, $0 \le \delta(\vec{n},\vec{n}') < 1$, being closer to 0 the greater is the similarity between $\vec{n}$ and $\vec{n}'$.

4.2.5 The BFB Count Vector Problem

With these definitions, we can now precisely pose a set of problems that need to be solved to evaluate copy number patterns for consistency with BFB:

BFB count vector problem variants

Input: *a count vector* $\vec{n} = [n_1, n_2, \dots, n_k]$.

- **The decision variant:** *decide if* $\vec{n}$ *is a BFB count vector.*
- **The search variant:** *if* $\vec{n}$ *is a BFB count vector, find a BFB string* α *such that* $\vec{n} = \vec{n}(\alpha)$.
- **The distance variant:** *Identify a BFB count vector* $\vec{n}'$ *such that* $\delta(\vec{n},\vec{n}')$ *is minimized. Output* δ.

4.3 Outline of the BFB Count Vector Algorithms

We defer the full details of the algorithms we have developed to Appendix C, presenting here only essential properties of BFB strings and some intuition of how to incorporate these properties in algorithms for BFB count vector problems. We focus on the search variant of the problem, where the goal of the algorithm is to output a BFB string α consistent with the input counts, if such a string exists.

4.3.1 Properties of BFB palindromes

Call an l-BFB string β of the form $\beta = \alpha\bar{\alpha}$ an *l-BFB palindrome*[1]. For an l-BFB string α, the string $\beta = \alpha\bar{\alpha}$ is an l-BFB palindrome by definition (choosing $\rho = \varepsilon$ and $\gamma = \alpha$ in Definition 2). In [36], it was shown that every prefix of a BFB string is itself a BFB string, thus, for an l-BFB palindrome $\beta = \alpha\bar{\alpha}$, the prefix α of β is also an l-BFB string. Hence, it follows that α is an l-BFB string if and only if $\beta = \alpha\bar{\alpha}$ is an l-BFB palindrome. For a BFB string α with $\vec{n}(\alpha) = [n_1, n_2, \ldots, n_k]$ and a corresponding BFB palindrome $\beta = \alpha\bar{\alpha}$, we have that $\vec{n}(\beta) = 2\vec{n}(\alpha) = [2n_1, 2n_2, \ldots, 2n_k]$. Thus, a count vector $\vec{n}$ is a BFB count vector if and only if there is a 1-BFB palindrome β such that $\vec{n}(\beta) = 2\vec{n}$. Considering BFB palindromes instead of BFB strings will facilitate the algorithm description.

Define an *l-block* as a palindrome of the form $\beta = \sigma_l\beta'\bar{\sigma}_l$, where β' is an $(l+1)$-BFB palindrome. For example, from the 4-BFB palindromes $\beta'_1 = \text{DE}\bar{\text{E}}\bar{\text{D}}\text{DE}\bar{\text{E}}\bar{\text{D}}$ and $\beta'_2 = \varepsilon$, we can produce the 3-blocks $\beta_1 = \sigma_3\beta'_1\bar{\sigma}_3 = \text{CDE}\bar{\text{E}}\bar{\text{D}}\text{DE}\bar{\text{E}}\bar{\text{D}}\bar{\text{C}}$ and $\beta_2 = \sigma_3\beta'_2\bar{\sigma}_3 = \text{C}\bar{\text{C}}$. It may be asserted that an l-block is a special case of an l-BFB palindrome. Next, we show how l-BFB palindromes may be decomposed into l-block substrings.

For a string $\alpha \neq \varepsilon$, denote by $top(\alpha)$ the maximum integer t such that σ_t or $\bar{\sigma}_t$ occur in α, and define $top(\varepsilon) = 0$. For two strings α and β, say that $\alpha \leq^t \beta$ if $top(\alpha) \leq top(\beta)$, and that $\alpha <^t \beta$ if $top(\alpha) < top(\beta)$. For example, for $\alpha = \text{AB}$ and $\beta = \text{ABCD}\bar{\text{D}}\bar{\text{C}}$, $top(\alpha) = 2$ and $top(\beta) = 4$, therefore $\alpha <^t \beta$.

[1]We assume that genomic segments σ satisfy $\sigma \neq \bar{\sigma}$, therefore strings of the form $\alpha\sigma\bar{\alpha}$ will not be considered palindromes.

Definition 3. *A string α is a* convexed l-palindrome *if $\alpha = \varepsilon$, or $\alpha = \gamma\beta\gamma$ such that γ is a convexed l-palindrome, β is an l-BFB palindrome, and $\gamma <^t \beta$.*

While every l-BFB palindrome α is also a convexed l-palindromes (since $\alpha = \varepsilon\alpha\varepsilon$), not every convexed l-palindromes is a valid BFB string. For example, $\alpha = $ A$\bar{\text{A}}$AB$\bar{\text{B}}\bar{\text{A}}A\bar{\text{A}}$ is a convexed 1-palindromes (choosing $\gamma = $ A$\bar{\text{A}}$, $\beta = $ AB$\bar{\text{B}}\bar{\text{A}}$), yet it is not a 1-BFB string. Instead, we have the following claim, proven in Appendix C:

Claim 1. *A string α is an l-BFB palindrome if and only if $\alpha = \varepsilon$, α is an l-block, or $\alpha = \beta\gamma\beta$, such that β is an l-BFB palindrome, γ is a convexed l-palindrome, and $\gamma \leq^t \beta$.*

From Definition 3 and Claim 1, it follows that an l-BFB palindrome α is a palindromic concatenation of l-blocks. In addition, for the total count $2n_l$ of σ_l and $\bar{\sigma}_l$ in α, α contains exactly n_l l-blocks, where each block contains one occurrence of σ_l and one occurrence of $\bar{\sigma}_l$. When n_l is even, α is of the form $\alpha = \beta_1\beta_2\ldots\beta_{\frac{n_l}{2}-1}\beta_{\frac{n_l}{2}}\beta_{\frac{n_l}{2}}\beta_{\frac{n_l}{2}-1}\ldots\beta_2\beta_1$, each β_i is an l-block. When n_l is odd, α is of the form $\alpha = \beta_1\beta_2\ldots\ \beta_{\lfloor\frac{n_l}{2}\rfloor}\ \beta_{\lfloor\frac{n_l}{2}\rfloor+1}\ \beta_{\lfloor\frac{n_l}{2}\rfloor}\ldots\beta_2\beta_1$. In the latter case, say that $\beta_{\lfloor\frac{n_l}{2}\rfloor+1}$ is the *center* of α, where in the former case say that the *center* of α is ε. Note that every l-block β appearing in α and different from its center occurs an even number of times in α. If the center of α is an l-block, this particular block is the only block which appears an odd number of times in α, while if it is an empty string then no block appears an odd number of times in α.

Now, let β be a 1-BFB palindrome with a count vector $\vec{n}(\beta) = 2\vec{n} = [2n_1, 2n_2, \ldots, 2n_k]$. It is helpful to depict β so that each character σ_l is at its own layer l, increasing with increasing l, as shown in Fig. 4.2a. As β is a concatenation of 1-blocks, we can consider the collection $B^1 = \{m_1\beta_1, m_2\beta_2, \ldots, m_q\beta_q\}$ of these blocks, where each count m_i is the number of distinct repeats of β_i in β. For example, for the string in Fig. 4.2a, $B^1 = \{2\beta_1, \beta_2, 2\beta_3, 4\beta_4\}$, where $\left|B^1\right| = n_1 = 9$, and β_2 is the center of β. Masking from strings in B^1 all occurrences of A and $\bar{\text{A}}$, each 1-block $\beta_i = \text{A}\beta_i'\bar{\text{A}}$ in B^1 becomes a 2-BFB palindrome β_i'. Such 2-BFB palindromes may be further decomposed into 2-blocks, yielding a 2-block collection B^2 (in Fig 4.2b, $B^2 = \{2\beta_5, \beta_6, 2\beta_7\}$, where $\left|B^2\right| = n_2 = 5$). In general, for each $1 \leq l \leq k$, masking in β all letters σ_r and $\bar{\sigma}_r$ such that $r < l$ defines a corresponding collection of l-block substrings of β. Each collection B^l

contains exactly n_l elements, as each l-block in the collection contains exactly two out of the $2n_l$ occurrences of σ_l in the string (where one occurrence is reversed). The collection B^{l+1} is obtained from B^l by masking occurrences of σ_l and $\bar{\sigma}_l$ from the elements in B^l, and decomposing the obtained $(l+1)$-BFB palindromes into $(l+1)$-blocks. We may define $B^{k+1} = \emptyset$ (where $\emptyset$ denotes an empty collection), since after masking in β all segments $\sigma_1, \ldots, \sigma_k$ we are left with an empty collection of $(k+1)$-blocks.

The algorithm we describe for the search variant of the BFB count vector problem exploits the above described property of BFB palindromes. Given a count vector $\vec{n} = [n_1, n_2, \ldots, n_k]$, the algorithm processes iteratively the counts in the vector one by one, from n_k down to n_1, producing a series of collections $B^k, B^{k-1}, \ldots, B^1$. Starting with $B^{k+1} = \emptyset$, each collection B^l in the series is obtained from the preceding collection B^{l+1} in a two-step procedure: First, $(l+1)$-blocks from B^{l+1} are concatenated in a manner that produces an $(l+1)$-BFB palindrome collection B' of size n_l (B' may contain empty strings, which can be thought of as concatenations of zero elements from B^{l+1}). Then, B^l is obtained by "wrapping" each element $\beta' \in B'$ with a pair of σ_l characters to become an l-block $\beta = \sigma_l \beta' \bar{\sigma}_l$. We will refer to the first step in this procedure as collection *folding*, and to the second step as collection *wrapping*. For example, in Fig 4.2d, the elements in $B^4 = \{4\beta_{10}\}$ are folded to form a 4-palindrome collection $B' = \{2\beta_{10}\beta_{10}, \varepsilon\}$ of size $n_3 = 3$. After wrapping each elements of B' by C to the left and $\bar{\text{C}}$ to the right, we get the 3-block collection $B^3 = \{2\text{C}\beta_{10}\beta_{10}\bar{\text{C}}, \text{C}\bar{\text{C}}\} = \{2\beta_8, \beta_9\}$. Algorithm SEARCH-BFB$(\vec{n})$ in Fig. 4.3 gives the pseudo-code for the described procedure, excluding the implementation of the folding phase which is kept abstract here. We next discuss some restrictions over the folding procedure, and point out that greedy folding is nontrivial. Nevertheless, in Appendix C we show an explicit implementation of a folding procedure, which guarantees that the search algorithm finds a BFB string provided that the input is a valid BFB count vector.

4.3.2 Required conditions for folding

Recall that the input of the folding procedure is an l-block collection B and an integer n, and the procedure should concatenate all strings in B in some manner to produce an l-BFB palindrome collection B' of size n. Since both l-blocks and empty

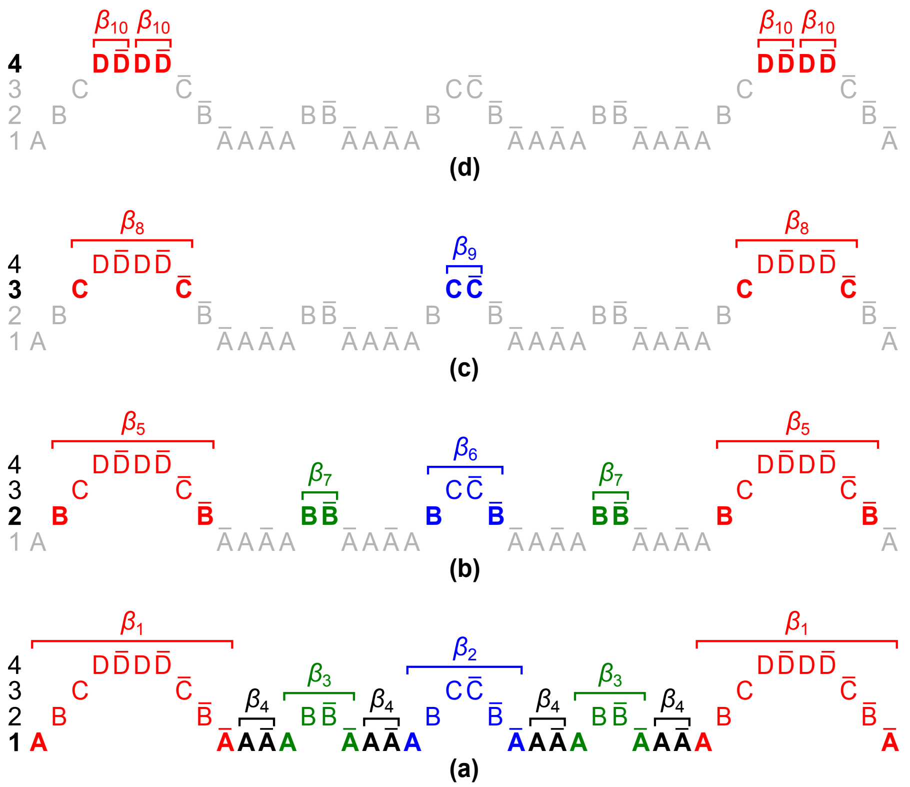


Figure 4.2: Layer visualization of a BFB palindrome $\beta = \alpha\bar{\alpha}$, where $\alpha = \mathrm{ABCD\bar{D}D\bar{D}\bar{C}\bar{B}\bar{A}A\bar{A}AB\bar{B}\bar{A}A\bar{A}ABC}$. A possible BFB sequence that produces α is $\mathrm{ABCD} \rightarrow \mathrm{ABCD}\mathbf{\bar{D}} \rightarrow \mathrm{ABCD\bar{D}}\mathbf{D\bar{D}\bar{C}\bar{B}\bar{A}} \rightarrow \mathrm{ABCD\bar{D}D\bar{D}\bar{C}\bar{B}\bar{A}}\mathbf{A} \rightarrow \mathrm{ABCD\bar{D}D\bar{D}\bar{C}\bar{B}\bar{A}A}\mathbf{\bar{A}AB} \rightarrow \mathrm{ABCD\bar{D}D\bar{D}\bar{C}\bar{B}\bar{A}A\bar{A}AB}\mathbf{\bar{B}\bar{A}A\bar{A}ABC}$. $\vec{n}(\alpha) = [9,5,3,4]$, and $\vec{n}(\beta) = 2\vec{n}(\alpha)$. Figures (a) to (d) depict layers 1 to 4 of β, respectively. In each layer l, the l-blocks composing the collection B^l are annotated as substrings of the form β_i. These collections are: $B^1 = \{2\beta_1, \beta_2, 2\beta_3, 4\beta_4\}$, $B^2 = \{2\beta_5, \beta_6, 2\beta_7\}$, $B^3 = \{2\beta_8, \beta_9\}$, $B^4 = \{4\beta_{10}\}$.

strings are special cases of l-BFB palindromes, when $n \geq |B|$ it is always possible to obtain B' by simply adding $n - |B|$ empty strings to B. Nevertheless, when $n < |B|$, there are instances for which no valid folding exists, as shown next.

For a pair of collections B and B', $B + B'$ is the collection containing all elements in B and B'. When $B'' = B + B'$, we say that $B = B'' - B'$ (note that $B'' - B'$ is well defined only when B'' contains B'). For some (possibly rational) number $x \geq 0$, denote by xB the collection $\{\lfloor xm_1 \rfloor \beta_1, \lfloor xm_2 \rfloor \beta_2, \ldots, \lfloor xm_q \rfloor \beta_q\}$. The operation $\text{mod2}(B)$ yields the sub-collection of B containing a single copy of each distinct element β with an odd count in B. For example, for $B = \{2\beta_1, \beta_2, 5\beta_3, 6\beta_4\}$, $\text{mod2}(B) = \{\beta_2, \beta_3\}$. Observe that $B = \text{mod2}(B) + 2\left(\frac{1}{2}B\right)$.

Claim 2. *Let B be an l-BFB palindrome collection such that* $\text{mod2}(B) = \emptyset$. *Then, it is possible to concatenate all elements in B to obtain a single l-BFB palindrome.*

Proof. By induction on the size of B. By definition, $\text{mod2}(B) = \emptyset$ implies that the counts of all distinct elements in B are even. When $B = \emptyset$, the concatenation of all elements in B yields an empty string ε, which is an l-BFB palindrome as required. Otherwise, assume the claim holds for all collections B' smaller than B. Let $\beta \in B$ be an element such that for every $\beta' \in B$, $top(\beta') \leq top(\beta)$, and let $B' = B - \{2\beta\}$. Note that $\text{mod2}(B') = \emptyset$ (since the count parity is identical for every element in both B and B'), and from the inductive assumption it is possible to concatenate all elements in B' into a single l-BFB palindrome α'. From Claim 1, the string $\alpha = \beta\alpha'\beta$ is an l-BFB palindrome, obtained by concatenating all elements in B. □

Claim 3. *Let B be an l-block collection. There is a folding B' of B such that $|B'| =$* $|\text{mod2}(B)| + 1$.

Proof. Recall that $B = \text{mod2}(B) + 2\left(\frac{1}{2}B\right)$. Since all element counts in the collection $2\left(\frac{1}{2}B\right)$ are even, $\text{mod2}\left(2\left(\frac{1}{2}B\right)\right) = \emptyset$, and from Claim 2 it is possible to concatenate all elements in $2\left(\frac{1}{2}B\right)$ into a single l-BFB palindrome α. Thus, the collection $B' = \text{mod2}(B) + \alpha$ is a folding of B of size $|\text{mod2}(B)| + 1$. □

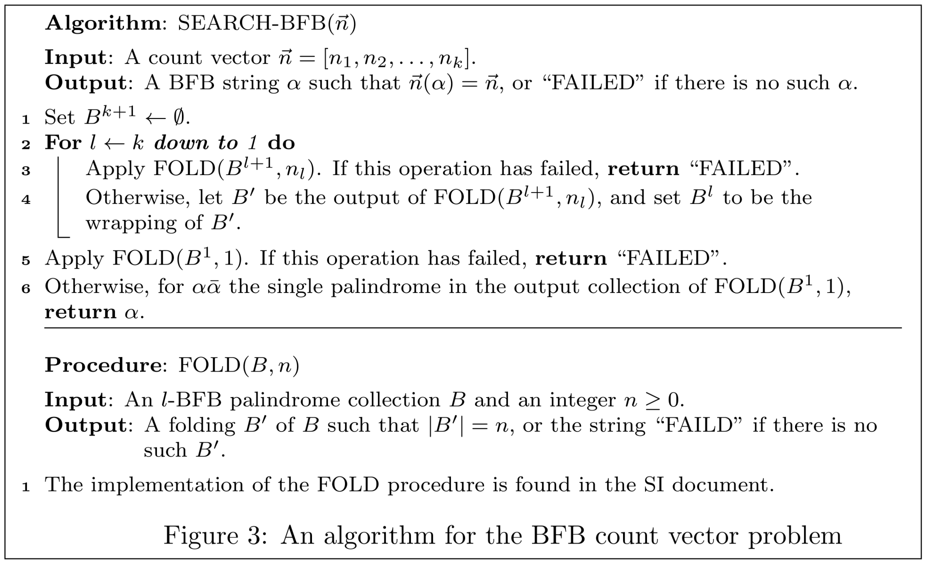
Claim 4. *For every folding B' of an l-block collection B,* $|\text{mod2}(B')| \geq |\text{mod2}(B)|$.

Proof. Let $\beta \in \text{mod2}\,(B)$ be an l-block repeating an odd number of times m in B. Therefore, β appears as a center of at least one element β' that occurs an odd number of times in B' (otherwise, β has an even number of distinct repeats as a substring of elements in B', in contradiction to the fact that m is odd). Hence, for each $\beta \in \text{mod2}\,(B)$ there is a corresponding unique element $\beta' \in \text{mod2}\,(B')$, and so $|\text{mod2}\,(B')| \geq |\text{mod2}\,(B)|$. □

The SEARCH-BFB$(\vec{n})$ algorithm described in Fig. 4.3 tries in each iteration l to fold the block collection B^{l+1} obtained in the previous iteration into an $(l+1)$-BFB palindrome collection of size n_l. When $n_l \geq \left|\text{mod2}\left(B^{l+1}\right)\right| + 1$, there always exists a folding as required: B^{l+1} maybe folded into a collection of size $\left|\text{mod2}\left(B^{l+1}\right)\right| + 1$ due to Claim 3, and additional $n_l - \left|\text{mod2}\left(B^{l+1}\right)\right| - 1$ empty strings may be added in order to get a folding of size n_l. On the other hand, when $n_l < \left|\text{mod2}\left(B^{l+1}\right)\right|$, no folding as required exists, due to Claim 4. In the remaining case of $n_l = \left|\text{mod2}\left(B^{l+1}\right)\right|$, the existence of an n_l-size folding of B^{l+1} depends on the element composition of B^{l+1}, as exemplified next.

Consider the run of Algorithm SEARCH-BFB$(\vec{n})$ over the input count vector $\vec{n} = [1,3,2]$. Here, $k = 3$, and the algorithm starts by initializing the collection $B^4 = \emptyset$. In the first loop iteration $l = 3$, and the algorithm first tries to fold the empty collection B^4 into a 4-BFB palindrome collection containing $n_3 = 2$ elements. Since there are no elements in B^4 to concatenate, the only way to perform this folding is by adding to B^4 two empty strings, yielding the collection $B' = \{2\varepsilon\}$, which after wrapping becomes $B^3 = \{2\text{C}\bar{\text{C}}\} = \{2\beta_1\}$. In the next iteration $l = 2$, and B^3 should be folded into a collection B' of size $n_2 = 3$. Among the possibilities to perform this folding are the following: $B'^a = \{2\beta_1, \varepsilon\}$, and $B'^b = \{\beta_1\beta_1, 2\varepsilon\}$, which after wrapping become $B^{2a} = \{2\text{B}\beta_1\bar{\text{B}}, \text{B}\bar{\text{B}}\} = \{2\beta_2, \beta_3\}$, and $B^{2b} = \{\text{B}\beta_1\beta_1\bar{\text{B}}, 2\text{B}\bar{\text{B}}\} = \{\beta_4, 2\beta_3\}$, respectively. Note that $\left|\text{mod2}\left(B^{2a}\right)\right| = \left|\text{mod2}\left(B^{2b}\right)\right| = 1$. Nevertheless, it is possible to fold B^{2a} in the next iteration into the collection $\{\beta_2\beta_3\beta_2\}$ of size $n_1 = 1$, while B^{2b} cannot be folded into such a collection. The reason is that the only concatenation of all elements in B^{2b} into a single palindrome is the concatenation $\beta_3\beta_4\beta_3$, but since $top\,(\beta_4) = top\left(\text{B}\text{C}\bar{\text{C}}\text{C}\bar{\text{C}}\bar{\text{B}}\right) = 3 > 2 = top\,(\text{B}\bar{\text{B}}) = top\,(\beta_3)$, Claim 1 implies that this concatenation is not a valid BFB palindrome.

In Appendix C, we define a property called the *signature* of a collection, and



```
Algorithm: SEARCH-BFB($\vec{n}$)
Input: A count vector $\vec{n} = [n_1, n_2, \dots, n_k]$.
Output: A BFB string $\alpha$ such that $\vec{n}(\alpha) = \vec{n}$, or "FAILED" if there is no such $\alpha$.
1 Set $B^{k+1} \leftarrow \emptyset$.
2 For $l \leftarrow k$ down to 1 do
3     Apply FOLD($B^{l+1}, n_l$). If this operation has failed, return "FAILED".
4     Otherwise, let $B'$ be the output of FOLD($B^{l+1}, n_l$), and set $B^l$ to be the
      wrapping of $B'$.
5 Apply FOLD($B^1, 1$). If this operation has failed, return "FAILED".
6 Otherwise, for $\alpha\bar{\alpha}$ the single palindrome in the output collection of FOLD($B^1, 1$),
  return $\alpha$.

Procedure: FOLD($B, n$)
Input: An $l$-BFB palindrome collection $B$ and an integer $n \geq 0$.
Output: A folding $B'$ of $B$ such that $|B'| = n$, or the string "FAILD" if there is no
        such $B'$.
1 The implementation of the FOLD procedure is found in the SI document.
```

Figure 3: An algorithm for the BFB count vector problem

Figure 4.3: An algorithm for the BFB count vector problem.

show how the exact minimum folding size depends on this signature. We also show how to fold a collection in a manner that optimizes this signature, and guarantees for valid BFB count vector inputs that the search algorithm finds an admitting BFB string.

4.4 Running time

For a count vector $\vec{n} = [n_1, \dots, n_k]$, let $N = \sum_{1 \leq i \leq k} n_i$ be the number of segments in a string corresponding to $\vec{n}$. Let $\tilde{N} = \sum_{1 \leq i \leq k} \log n_i$ denote a number proportional to the number of bits in the representation of $\vec{n}$, assuming each count n_i is represented by $O(\log n_i)$ bits. In Appendix C, we complete the implementation details of algorithms for the decision, search, and distance variants of the BFB count vector problem, and show these algorithms have the asymptotic running times of $O(\tilde{N})$ (bit operations), $O(N)$, and $O(N^{\log N})$ (under some realistic assumptions), respectively. For the decision and search variants, these running times are optimal, being linear in the input (for the decision variant) or output (for the search variant) lengths.

In practical terms, this has a significant effect on our ability to evaluate copy number signatures of BFB when compared to the previous exponential-time algorithm [36]. To determine if a count vector consistent with BFB is in fact strong evidence for

BFB, we have to check many count vectors. Analyzing the simulations we explain below required testing tens of millions of different count vectors, so even a small improvement in running time can have a large impact of the scope of analysis we can perform.

But, the running time improvement with the new algorithm is not small. For example, a count vector that took 9 seconds with the previous algorithm can be processed by the new algorithm in $1.2\text{x}10^{-5}$ seconds. A count vector that needed 148 seconds with the old algorithm now completes in $1.9\text{x}10^{-5}$ seconds. A count vector that was abandoned after 30 hours with the old algorithm now takes only $8.1\text{x}10^{-6}$ seconds. Thus, the improvement in running time is not of merely theoretical interest. The earlier algorithm did not allow a thorough study of longer count vectors, while with the new algorithm such a study is possible.

4.5 Detecting Signatures of BFB

We can now describe the two features we will use to determine if a chromosome has undergone BFB. The first feature is based on the fold-back inversions that BFB produces. For a given region, we can find all the breakpoints identified by sequencing and determine what proportion are fold-back inversions. We call this the *fold-back fraction*. The second feature relies on our algorithm that solves the BFB count vector problems we have posed. For a given contiguous pattern of copy counts, that is, a count vector, we can find the distance to the nearest count vector that could be produced by BFB using the distance metric we defined above. We call this the *count vector distance*. For a particular count vector, we define a score s that combines these two features:

$$s = \lambda\delta + (1-\lambda)(1-f) \tag{4.1}$$

Here, f refers to the fold-back fraction, δ refers to the count vector distance, and λ refers to the weight we give to the count vector distance versus the fold-back fraction when calculating the score. When $\lambda = 1$, we are only looking at count vector distance, whereas when $\lambda = 0$, we are only using fold-back fraction and ignoring the count vectors.

4.6 Results

To determine whether our two proposed features could identify BFB against the complex backdrop of a cancer genome, we simulated rearranged chromosomes. Our overall goal was to simulate cancer chromosomes that were highly rearranged yet had not undergone BFB to see if evidence for BFB appeared in them, suggesting that using such evidence would lead to false positives. Conversely, we also wanted to simulate chromosomes whose rearrangements included BFB to determine if a proposed BFB signature was sensitive enough to identify BFB when it occurred. Since it is not clear how to faithfully simulate cancer genome rearrangements, we used a wide range of simulation parameters so we could understand how different assumptions affect the features' ability to identify BFB.

We began with a pair of unrearranged chromosomes and then introduced 50 rearrangements to each. Each rearrangement was an inversion, a deletion, or a duplication. Duplications were either direct or inverted and could be tandem or interspersed. The type of each rearrangement was chosen from a distribution. In some chromosome pairs, we imitated BFB by successively duplicating and inverting segments of one end of one chromosome for each round of BFB. The number of BFB rounds varied from two to ten. Then, we calculated the copy counts and breakpoints for the chromosome pair and introduced error to the copy counts according to a random model and also randomly deleted or inserted breakpoint observations. For each combination of rearrangement type distribution and number of BFB rounds, we simulated 5,000 chromosome pairs with BFB and 15,000 without BFB. Complete details are in Appendix C.

We first examined the usefulness of count vector distance alone in identifying BFB by setting $\lambda = 1$ in our score function (Eqn. 4.1). For each chromosome pair, we found all contiguous count vectors of a given length and calculated their scores, as described above and in Appendix C. We used the minimum score s over all of these sub-vectors in the chromosome as a score for the whole chromosome. Then, for varying thresholds, we classified all chromosomes with a score lower than the threshold as having been rearranged by BFB. The performance of this classification varied with the parameters used to simulate the chromosomes, but typical results can be seen in Figure 4.4a. The solid lines show ROC curves for different count vector lengths for the simula-

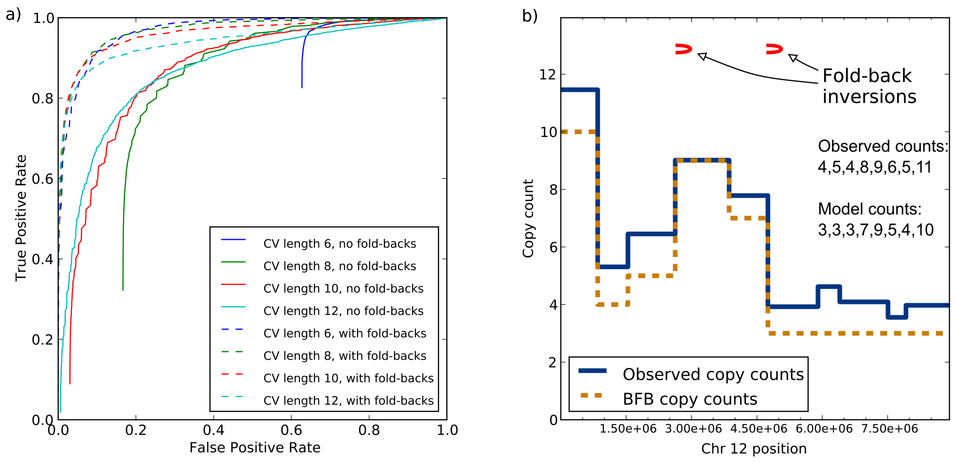


Figure 4.4: Simulation and pancreatic cancer results. a) ROC curves for different count vector lengths with and without fold-back fractions for the simulation of eight BFB rounds and equally likely other rearrangements. b) Observed copy counts and copy counts compatible with BFB on the short arm of chromosome 12 in pancreatic cancer sample PD3641. The presence of fold-back inversions and the count vector's consistency with BFB suggests that this portion of chromosome 12 underwent BFB cycles.

tion with eight rounds of BFB and a distribution that yields roughly equal probabilities of the other rearrangement types. Consistent with previous observations, short count vectors that are perfectly consistent with BFB can be found in many chromosomes, even if BFB did not occur. So, even with a score threshold of zero, they would still be classified as consistent with BFB. For example, 63% of chromosomes without any true BFB rearrangements in Figure 4.4a had a count vector of length six perfectly consistent with BFB.

In contrast, examining longer count vectors produced a better classification. For instance, setting the score threshold to .10, count vectors of length twelve could achieve a true positive rate (TPR) of 67% and a false positive rate (FPR) of only 10%. However, this performance must be considered in the context of an experiment seeking evidence for BFB. Chromosomes that have undergone BFB are probably rare. If only one in a hundred chromosomes tested underwent BFB, then a test with an FPR of even 1% will produce mostly false discoveries. Achieving this FPR with count vectors of length twelve with the chromosomes in Figure 4.4a would result in a TPR of only 16%. A more appropriate target FPR for screening many samples, say .1%, could not be achieved with count vectors alone.

Next, we incorporated fold-back inversions into the scoring function. We set $\lambda = .5$, giving equal weight to fold-back fraction and count vector distance. ROC curves using this approach are shown by dashed lines in Figure 4.4a. Incorporating fold-back fractions into the scoring leads to better discrimination of chromosomes with and without BFB rearrangements; the test in Figure 4.4a that combines count vectors of length 12 and fold-back inversions can achieve a TPR of 48% with an FPR of .1% by setting the score threshold to .27. This suggests that it could detect BFB in a large dataset without being overwhelmed by false discoveries.

Of course, these conclusions depend on our simulation resembling actual cancer rearrangements and BFB cycles. A true specification of cancer genome evolution is unknown and in any case varies from cancer to cancer. Recognizing this complication, we repeated the analysis in Figure 4.4a for the different rearrangement distributions, number of BFB rounds, and count vector lengths. For each combination, we recorded the score threshold needed to achieve FPRs of .1%, 1%, and 5%, and the respective expected TPRs. The full results are shown in Dataset S1 and ROC curves are shown in Figures C.1-C.5. Generally, different simulations showed the same trends. Fold-back inversions alone were better at identifying BFB than count vectors alone, but the combination of both features provided the best classification. By examining a wide range of simulation parameters, we illustrate how changes in assumptions about cancer genome evolution and BFB influence the appropriateness and expected outcomes of tests for BFB.

We applied our method to a publicly available dataset of copy number profiles from 746 cancer cell lines [6]. We found three chromosomes with count vectors of length 12 nearly consistent with BFB: chromosome 8 from cell line AU565, chromosome 10 from cell line PC-3, and chromosome 8 from cell line MG-63 (see Appendix C). While the patterns of copy counts on these chromosomes do bear the hallmarks of BFB, our simulations suggest that labeling chromosomes as having undergone BFB based on these count vectors would lead to an FPR between 1% and 10%. Given that thousands of chromosomes were examined, many of which were highly rearranged, the consistency of these copy counts with BFB may be spurious.

We also applied our method to paired-end sequencing data from seven previously published pancreatic cancer samples [9]. We estimated copy numbers from the reads and used breakpoints as reported by the original investigators. We examined count vectors of length 8 and chose a threshold score of .18, which would give an FPR of .1% based on simulations where the non-BFB rearrangement types are roughly equally likely. We identified two chromosomes that showed evidence for BFB, both from the same sample, PD3641. The first was the long arm of chromosome 8. This chromosome was identified by the original investigators as likely being rearranged by BFB. Our analysis suggests that, barring rearrangements that differ significantly from any of our simulations, this chromosome did indeed undergo BFB cycles. We also found evidence for BFB rearrangements from a count vector spanning ten megabases on the short arm of chromosome 12 (Figure 4.4b). Thus, we were able to recover evidence for BFB previously identified by hand curation. And by combining count vector and fold-back analysis, we found an additional strong BFB candidate that would not be apparent without modeling and simulation.

4.7 Discussion

Some 80 years after Barbara McClintock's discovery of the Breakage Fusion Bridge mechanism, it is seeing renewed interest in the context of tumor genome evolution. Recent publications have claimed, based on empirical observations of segmentation counts and other features, that their data counts are "consistent with BFB". The main technical contribution of the chapter is an efficient algorithm for detecting if given segmentation counts can indeed be created by Breakage Fusion Bridge cycles. That algorithm turns out to be non-trivial, requiring a deep foray into the combinatorics of BFB count vectors, even though its final implementation is straightforward and fast. Experimenting with the implementation reveals that in fact, (a) there is a big diversity of count-vectors created by true BFB cycles not all of which are easily recognizable as BFB; and, (b) at least for short count-vectors, it is often possible to create BFB-like vectors by non-BFB operations. Thus, being "consistent with BFB", and "caused by BFB" are not equivalent. Fortunately, our results also suggest that using longer count vectors, and additional information of fold-backs gives stronger prediction of BFB, even in the presence of noise, and diploidy. While assembly of these highly rearranged genomes continues to be difficult, recent advances in long single-molecule sequencing will provide additional spatial information that will improve the resolving power of our algorithm. As more cancer genomes are sequenced, including single-cell sequencing, the method presented here will be helpful in determining the extent and scope of BFB cycles in the evolution of the tumor genome.

4.8 Acknowledgements

This research was supported by grants from the National Institute of Health (5RO1-HG004962, U54 HL108460) and the National Science Foundation (NSF-CCF-1115206).

Chapter 4 (with Appendix C) was published in the *Proceedings of the National Academy of Sciences of the United States of America*, 2013, S. Zakov, M. Kinsella, and V. Bafna, “An algorithmic approach for breakage-fusion-bridge detection in tumor genomes.” The dissertation author was a primary co-investigator and co-author of this paper.

Chapter 5

Does Chromothripsis Have a Distinguishing Signature?

5.1 Introduction

In a groundbreaking study 2011 study [72], Stephens *et al.* observed a pattern of structural variation in a leukemia genome so atypical it presumptively revealed a novel mechanism of chromosome rearrangement. Two features distinguish this variation pattern. First, the chromosome or chromosomal region in question has many clustered breakpoints that suggest complex adjacencies rather than simple deletions or non-overlapping tandem duplications. Second, the region oscillates between two or perhaps three copy number states.

To further investigate this phenomenon, Stephens *et al.* sequenced several cell lines with chromosomes that exhibited these features. One of these chromosomes was chromosome 15 from SNU-C1, a colon cancer cell line. This chromosome has 239 breakpoints identified by paired-end sequencing (PES) and mostly oscillates between two copy number states, two and four. Using simulations, Stephens *et al.* showed that the progressive introduction of the breakpoints they observed would result in a chromosome with many copy number states rather than just two. They hypothesized that the peculiar rearrangement pattern was not the result of progressive rearrangements but instead the result of the chromosome shattering followed by the random stitching together of the resulting pieces. They termed this phenomenon “chromothripsis”.

To determine how widespread chromothripsis may be, Stephens *et al.* used the progressive rearrangement simulation from SNU-C1 to conclude that a chromosome with at least 50 breakpoints dominated by at most three copy number states was unlikely to have been rearranged progressively and thus was likely to be a product of chromothripsis. Using these criteria they searched copy number profiles and found 2-3% of cancers have a chromosome that bears the hallmark of chromothripsis.

This is a striking result; it suggests a mechanism of cancer genome evolution that contrasts starkly with previously described models. This discovery has generated excitement and ongoing investigation. Subsequent studies have found evidence for chromothripsis in multiple myeloma [47], medulloblastoma [63, 57], neuroblastoma [55], and colorectal cancers [39] as well as the germline [38, 14]. Moreover in some studies, chromothripsis has been associated with more aggressive cancers. Thus, it would appear that a new source of human disease has been found, with potentially far-reaching effects on our understanding and treatment [58] of cancer.

The great potential of chromothripsis cannot be realized unless it can be accurately detected. It is unlikely that chromothripsis will ever be reliably observed directly, so we will need to rely on the footprint that chromothripsis should leave in copy number and breakpoint data. The characterization of this footprint is an open problem. While Stephens *et al.* searched for chromosomes dominated by at most three copy number states with at least 50 positions where copy number changes, subsequent works have used more relaxed criteria. They have required fewer breakpoints per chromosome, such as 20 [55], 10 [63, 57], or just a handful [14]. They also have not always required that the number of unique copy states in a chromosome be limited to two or three [57, 55].

The validity of these footprints of chromothripsis rests on the idea that progressive rearrangement cannot create such patterns. However, the evidence for this proposition is largely limited to the initial simulation work by Stephens. Chromothripsis is now being investigated in different contexts than Stephens’ cell line simulations. Furthermore, the diversity of approaches used to identify chromothripsis means some groups are likely over- or underestimating its prevalence. This, together with the potentially great significance of chromothripsis, highlights the value of revisiting and extending the simulation work that underlies current strategies for identifying chromothripsis.

In this chapter, we review the simulation approach that suggests that progressive rearrangements cannot yield a chromosome with many breakpoints and few unique copy number states. First, we explore whether changes to the implementation of the simulation affects the validity of the footprint of chromothripsis. We show that a subtle but consequential error in the original implementation of the simulation causes it to understate the breakpoint and copy number patterns that can be achieved by progressive rearrangement. We examine varying possible meanings of "breakpoint" and "copy number state" and determine definitions that more closely correspond to experimental results. Next, we show that progressive rearrangement with a preference for inversions can produce chromosomes that bear the putative footprint of chromothripsis. Together these issues suggest that, assuming the simulation approach is valid, more stringent criteria must be used to identify chromothripsis and that the current literature overstates its prevalence.

We then demonstrate that the simulation approach produces similar results whether a chromosome is progressively rearranged or not. This undermines its ability to distinguish between chromothripsis and progressive rearrangement. Extending on this finding, we demonstrate a method that finds plausible progressive rearrangements that explain the breakpoints of particular chromosomes that appear to have undergone chromothripsis. Finally, we offer a discussion of the significance of these findings for the chromothripsis hypothesis.

5.2 Methods

5.2.1 Finding Chromosome Arrangements Consistent with Observed Breakpoints

The input to the method is a set of breakpoints, {(Pos1, Strand1, Pos2, Strand2) . . . }, where Pos1 and Pos2 correspond to the unexpectedly adjacent positions in the chromosome and Strand1 and Strand2 give the orientations of the chromosome at each position. If we collect each Pos1 and Pos2 from each breakpoint and sort them, we get the locations of all breakpoint boundaries in the chromosome ordered from one end of the chromosome to the other. We can consider the chromosomal intervals that lie between subsequent breakpoint boundaries as the segments of the chromosomes that end up being rearranged.

Our goal in to find an ordering of the segments such that the segment adjacencies correspond to the observed breakpoints and that no chromosomal segment appears more than once. To do this, we create a graph. The vertices of the graph are breakpoint boundaries plus two additional vertices for the start and end of the chromosome. There are two types of edges. The first type, "segment edges", correspond to the chromosome segments. The second type, "breakpoint edges", correspond to the breakpoints. We want a path through the graph that crosses as many breakpoint edges as possible without crossing any segment edge more than once. The path must also correspond to a real chromosome, so there are a number of other restrictions like two breakpoint edges cannot be traversed in a row and the direction a segment edge can be traversed is determined by the previous breakpoint or segment edge.

5.3 Results

5.3.1 Simulating Progressive Rearrangements

We will first summarize the simulation method. Consider a chromosome 100 bases long that undergoes chromothripsis, shattering into ten segments of ten bases, which we label A through J. The segments come back together, but some are lost, some are inverted, and the order is shuffled. Suppose the resulting arrangement of segments is AE(-C)(-G)HI. If this chromosome is sequenced, it will reveal five breakpoints and copy numbers that alternate between zero and one. The positions and orientations of the breakpoints are shown in Table 5.1, and an illustration of the chromosome is shown in Figure 5.1.

We can now step through the progressive rearrangement simulation used by Stephens *et al.* The simulated chromosome begins intact, with no rearrangements (Figure 5.2a). Then, a random breakpoint is chosen from the set of observed breakpoints. In this case, suppose the breakpoint between 20 and 70 is chosen. This breakpoint is now

Table 5.1: Breakpoint positions and orientations for rearranged chromosome in Figure 5.1.

Lower Position	Orientation at Lower Position	Higher Position	Orientation at Higher Position
10	+	40	+
30	+	50	-
20	-	70	-
60	-	70	+
80	+	90	+

introduced into the chromosome via one of three rearrangement types: inversion, deletion, or tandem duplication. The observed orientation of the two ends of the breakpoint is - -. So, an inversion cannot be used to create the breakpoint because that will result in segments with orientations of + - or - +. A deletion between 20 and 70 will not work because then the orientations would be + +. But, a tandem duplication between 20 and 70 will result in a breakpoint that, when read from 20 to 70, will have both segments in reversed orientation. So, segments C through G are duplicated. Next, the rearrangement between 30 and 50 is chosen. Using similar reasoning as above, this rearrangement is introduced via an inversion of segments FGC. Note that this creates two breakpoints, the observed breakpoint plus another one in opposite orientation. Then, two more rearrangements are introduced resulting in the chromosome in Figure 5.2e. The number of breakpoints and copy number states in this chromosome would be recorded, and the simulation would be repeated many times with different rearrangement orders and segment choices. It would also be stopped when varying numbers of breakpoints had been introduced so that the relationship between the number of breakpoints and the number of unique copy number states could be determined.

Stephens *et al.* graciously shared the code they used to produce their results. We have reimplemented the method, applied it to chromosome 15 of SNU-C1, and replicated their results (Figure 5.3a). The general trend, consistent with Stephens' result, is that the number of unique copy number states increases with the number of breakpoints. A chromosome with 239 breakpoints and only two copy number states falls well outside

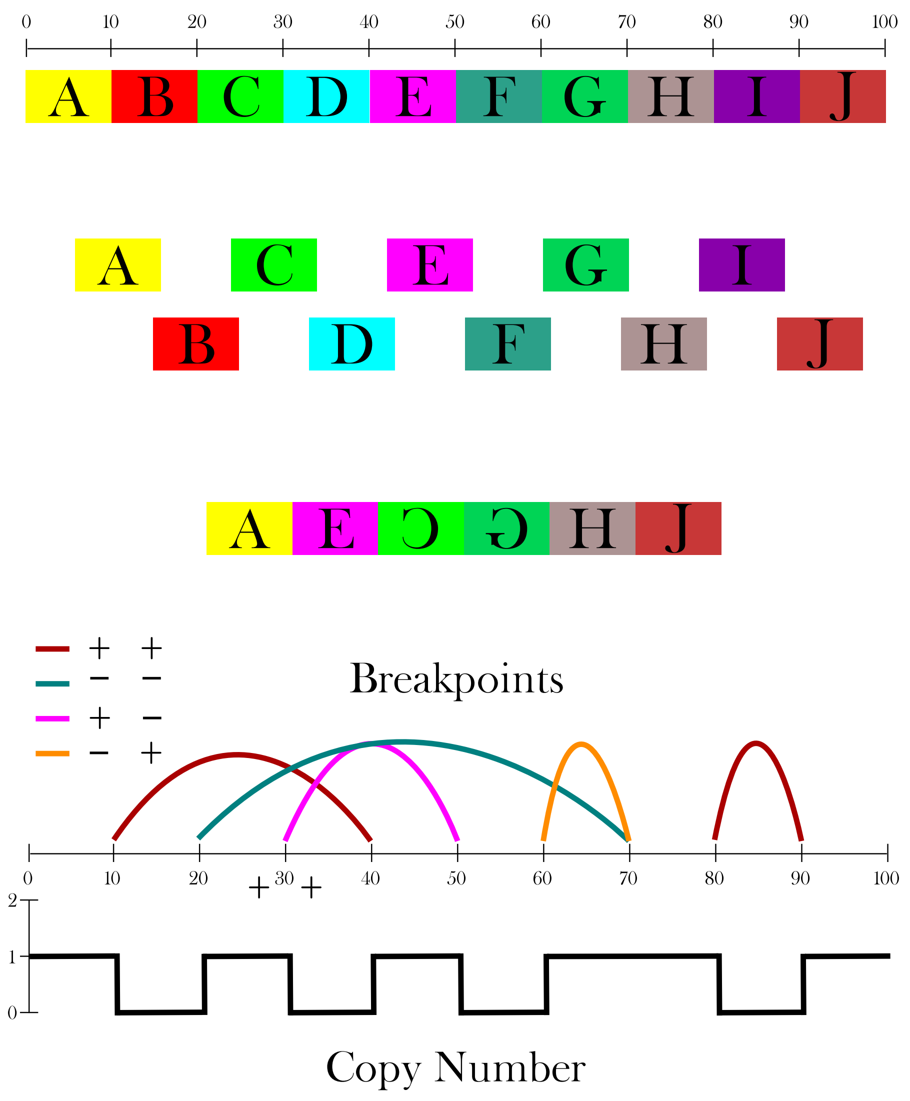


Figure 5.1: A hypothetical shattered chromosome.

of what was produced by the progressive simulation, and this is a key piece of evidence that chromosome 15 of SNU-C1 is the result of chromothripsis rather than progressive rearrangement. Moreover, based on the chart, it appears that a chromosome with at most three copy number states and more than fifty, or perhaps even twenty, breakpoints also falls outside of what can be achieved by progressive rearrangement.

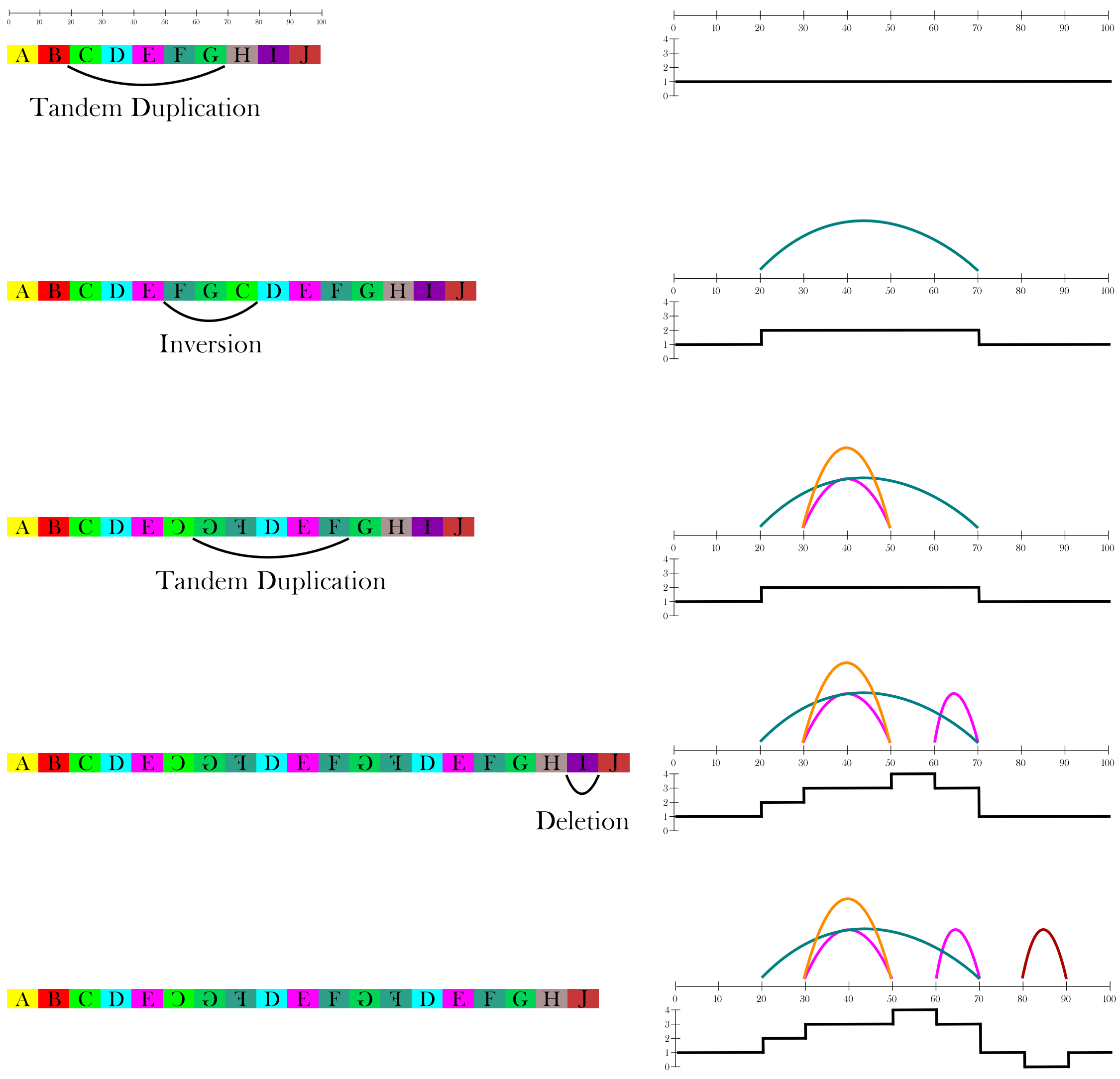


Figure 5.2: A set of possible simulation steps.

5.3.2 Chromothripsis Footprint Criteria Depend on Subtle Simulation Implementation Details

The above result is more meaningful if it is robust to changes in the implementation of the simulation. In this section, we alter the simulation in various ways to determine if the proposed footprint of chromothripsis remains valid when assumptions about progressive rearrangement are changed.

The first change we made to the simulation was a correction of a logic error that caused some simulated inversions to behave like duplications. The details are in the supplement, but the net effect was that some operations that ought to have preserved existing copy numbers instead introduced up to two new copy number states to the chromosome. When we corrected this, the chart of copy number states and breakpoints shifted down (Figure 5.3b). This change in result does not affect inferences about chromosome 15 of SNU-C1, since 239 breakpoints and only two copy number states is still well outside of the simulated results. But, the simulated chromosomes now begin to encroach upon the chromothripsis region of the graph. For example, the new simulation produced a chromosome with 67 breakpoints and only 3 copy number states, which is consistent with the footprint of chromothripsis even though the chromosome was rearranged progressively.

The next alteration was to the counting breakpoints and copy number states. Thus far, we have been imprecise about the meaning of the breakpoint values on the x-axes of our charts. This imprecision is also found in the literature, but there are in fact multiple ways to count breakpoints on a chromosome. One way is to count the number of times an abnormal adjacency appears. For example in the chromosome in Figure 5.2e, moving from left to right we find six such adjacencies: E(-C), (-C)(-G), (-F)D, F(-G), (-F)D, and HJ. This counting method was used in Figures 5.3a and 5.3b. Another way to count breakpoints is to consider how the breakpoints would be reported by a PES experiment. This is similar to the previous method, except that if an abnormal adjacency appears in the chromosome multiple times because of duplications, it will only appear once in the sequencing results. So referring back to Figure 5.2e, the adjacency (-F)D would only be counted once even though it appears in the chromosome twice. A third way to count breakpoints is to consider how they will appear in a microarray or depth of coverage experiment. This method counts breakpoints where copy number changes. The copy numbers in the chromosome in Figure 5.2e from left to right are 1,2,3,4,3,1,0,1. So, copy number changes seven times.

There are also multiple ways to count the number of copy number states in a chromosome. The first we can call “strict”. With this method, we simply count the number of copy number states observed in the chromosome, regardless of how much of the chromosome is covered by any copy number state. In Figure 5.2e, there are five copy states observed, zero through four. Another method, which we will call “relaxed”, counts how many copy states are needed to cover some fraction of the chromosome.

If we use the fraction 90%, then the relaxed number of copy states in the chromosome above is four because we can cover 90 bases using only four copy number states. Relaxed counting of copy states can be appropriate for identifying chromothripsis because it allows us to find chromosomes that are dominated by two or three copy number states but may have some small regions with other copy numbers because of subsequent alterations or experimental error.

The simulation by Stephens *et al.* used strict copy number state counting and the first breakpoint counting method, counting every unexpected adjacency even if duplicated. In contrast, the breakpoints observed in chromosome 15 of SNU-C1 come from PES, and the copy number state count of two was arrived at using relaxed counting. Microarray results show that the chromosome has six copy number states using strict counting [34].

We modified the simulation to use relaxed copy state counting that found how many copy number states were needed to cover 95% of the simulated chromosome. When this was combined with PES breakpoint counting, it produced the results in Figure 5.3c; when combined with microarray breakpoint counting, it produced Figure 5.3d. Because of the changes in breakpoint counting, the simulations could no longer quickly produce chromosomes with over 100 breakpoints. Both simulation also showed a continuation of the trend seen in Figure 5.3b with a narrowing separation between the simulated chromosomes and chromosomes bearing the footprint of chromothripsis. For example, of the 414 chromosomes in Figure 5.3c with between 50 and 55 breakpoints, 16 (3.9%) were dominated by three or two copy number states. This suggests that in a screen of many chromosomes, the proposed footprint of chromothripsis may produce false discoveries.

Finally, we altered the way the simulation chooses breakpoints to introduce into the chromosome. In the original simulation, breakpoints were chosen uniformly randomly without replacement, so each remaining breakpoint had an equal chance of being introduced at each step. This may not correspond to biological reality as there may be some preference for particular kinds of rearrangements. Specifically, a preference for inversions over other rearrangement types could lead to chromosomes with many breakpoints but few copy number states. To test this, we changed the simulation so that inversions were twice as likely to be chosen at each step compared to deletions or duplications. The results are in Figures 5.4a and 5.4b, using PES and microarray breakpoint counting respectively. These results have many simulated chromosomes bearing the footprint of chromothripsis. The large fraction of chromosomes with many breakpoints and few copy number states (Table 5.2) indicates that some chromosomes that appear to have undergone chromothripsis could also have been produced by progressive rearrangement that favors inversions.

Table 5.2: Fraction of chromosomes in Figure 5.4a with few copy number states for given breakpoint counts.

Breakpoint Range	**Fraction of Chromosomes With 2 or 3 Copy Number States**
50-59	12.6%
60-69	7.4%
70-79	2.6%
80-89	0.8%
90-99	0.6%

The results in this section suggest that a more conservative threshold should be used to identify chromothripsis in order to avoid false discoveries. If the minimum number of breakpoints were set at 100 rather than 50, much of the risk of false discovery we have demonstrated above would be diminished. However, this threshold would also decrease the estimate of the prevalence of chromothripsis. When Stephens *et al.* screened 746 cancer cell line copy number profiles for chromosomes with over 50 breakpoints and at most three copy number states, they found chromosomes from 18 cell lines that met these criteria. With a threshold of 100 breakpoints, the number of cell lines drops to 3. So based on this analysis, the true prevalence of chromothripsis may be less than .5% rather than the original estimate of 2-3%.

5.3.3 Simulation Method Does Not Distinguish Between Progressive Rearrangement and Chromothripsis

In the previous section, we discussed implementation details of simulations of progressive rearrangements. We now turn our attention to the question of whether such simulations can provide reliable evidence for chromothripsis at all. In order for an experiment to provide information about a hypothesis, it has to produce different results when the hypothesis is true than when it is false. In order for simulations to demonstrate whether a chromosome could have been rearranged progressively, the simulations should produce different results for progressively rearranged chromosomes and chromosomes that have undergone chromothripsis.

The footprint of chromothripsis, many breakpoints with few unique copy states, is unlikely to appear in a chromosome rearranged by progressive and overlapping tandem duplications. However, it may appear in a chromosome rearranged by progressive inversions and deletions. We simulated such a chromosome with only inversions and deletions. The resulting breakpoints and copy numbers are shown in Figure 5.5. Even though only two kinds of rearrangements were used, the chromosome shows the same complex rearrangement pattern seen in chromosomes that have putatively undergone chromothripsis.

We then applied the simulation method to the breakpoints of this chromosome and recorded the results as we did in Figure 5.3b. The resulting distribution of breakpoints and copy number states in Figure 5.6 is not different from Figure 5.3b even though we know the chromosome was rearranged progressively. This result casts doubt on the usefulness of the simulation method to detect chromothripsis. Rather than distinguishing between chromosomes that shattered and chromosomes that were rearranged progressively, it always report that chromosomes with many complex rearrangements and few copy number states are the product of chromothripsis even when they are not.

5.3.4 Plausible Progressive Rearrangement Schemes Exist for Chromosomes Bearing Footprint of Chromothripsis

Thus far, we have discussed in general whether some chromosomes that appear to be the product of chromothripsis may actually have been progressively rearranged. We now move from the general to the specific to see if we can find series of progressive rearrangements that explain particular chromosomes that bear the footprint of chromothripsis. Stephens *et al.* singled out three chromosomes from three different cell lines for extensive sequencing and analysis: chromosome 5 from TK10, chromosome 9 from 8505C, and chromosome 15 from SNU-C1. These chromosomes had 55, 77, and 239 breakpoints respectively and oscillated between two copy number states. We want to find rearrangements that produce roughly the same breakpoints as were observed and keep the number of copy states at two in each chromosome.

To do this, we searched for sequences of inversions and deletions that yielded the breakpoints. By only using these two rearrangement types, we could guarantee that the copy number of every segment in the chromosome was zero or one. We found these sequences by first finding orderings of chromosomal segments with similar breakpoint profiles to those observed experimentally. Then, we used a tool called GRIMM[75] to find inversions and deletions that would create the orderings (see Methods). For each of the three chromosomes, we were able to discover series of inversions and deletions that yielded ~95% of the experimentally observed breakpoints as well as some additional breakpoints beyond what was observed (Table 5.3). Figure 5.7 illustrates the result for chromosome 5 from TK10. Animations of the series of rearrangements for each of the three chromosomes are in the supplement.

These series of progressive rearrangements raise potential alternative hypotheses for the complex breakpoints and oscillating copy number states in these chromosomes. Thus, while these chromosomes may have indeed undergone chromothripsis, the observations can also be explained using progressive rearrangements alone.

Table 5.3: The number of observed breakpoints.

Cell Line	Experimental Breakpoints	Covered Breakpoints	New Breakpoints
TK10	55	52	12
8505C	77	74	19
SNU-C1	239	228	75

5.4 Discussion

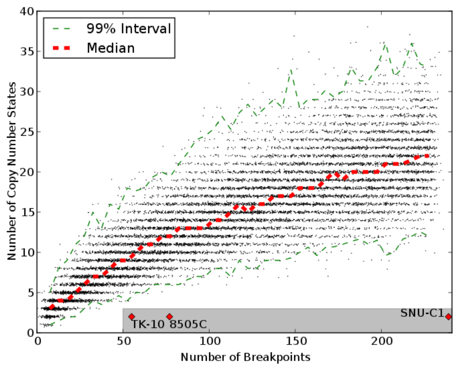
It is notoriously difficult to make sense of many cancer genomes due to the complexity of rearrangements. The proposal of the chromothripsis hypothesis was an important step forward as a possible mechanism for the creation of this complexity. Careful investigation of the phenomenon may deepen knowledge of structural variation in cancers.

At the same time, the proposal of ‘shattering and subsequent reassembly’ of a chromosome in a small number of cellular generations is truly extraordinary. The invocation of chromothripsis to explain molecular data from cancer samples must be done with great circumspection, and caution, even. The case for chromothripsis rests on the argument that there are some variation patterns that progressive rearrangement cannot achieve. But in this chapter, we have shown that progressive rearrangements can indeed achieve patterns that, at first glance, would seem quite unlikely. We demonstrated that a previously asserted footprint of chromothripsis may in fact encompass chromosomes rearranged progressively, that simulations might always rule out progressive rearrangement regardless of how the chromosome truly evolved, and that it is possible to find progressive rearrangements that explain chromosomes that appear to be exemplars of chromothripsis.

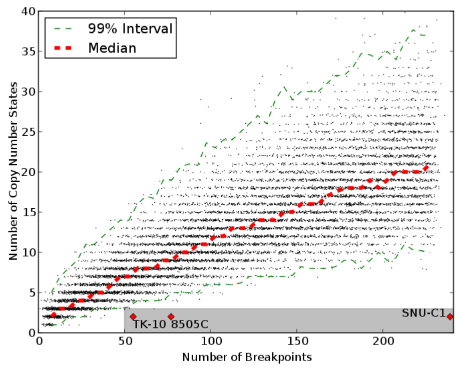
These results do not foreclose upon the chromothripsis hypothesis, of course. But, they do underscore difficulty of making inferences about mechanisms in cancer. Future work will likely refine the footprint of chromothripsis, but until then simply examining counts of breakpoints and copy states will provide only a limited understanding of the mechanisms underlying complex rearrangements.

5.5 Acknowledgements

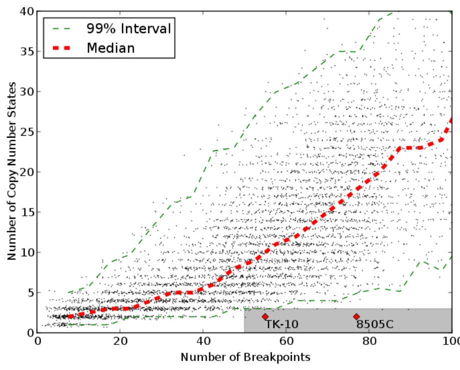
Chapter 5 is currently in submission, M. Kinsella, A. Patel, and V. Bafna, “Does Chromothripsis Have a Distinguishing Signature”. The dissertation author was the primary investigator and author of this paper.



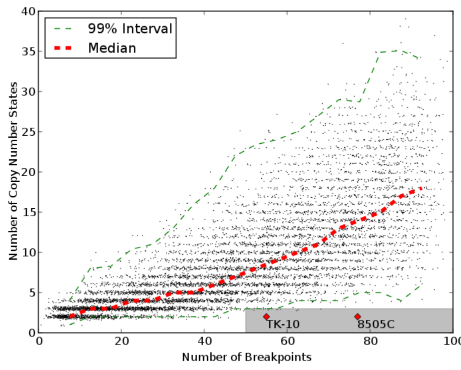
a. Results of directly reimplementing the simulation method of Stephens *et al*.



b. Results after fixing indexing issue for inversions.

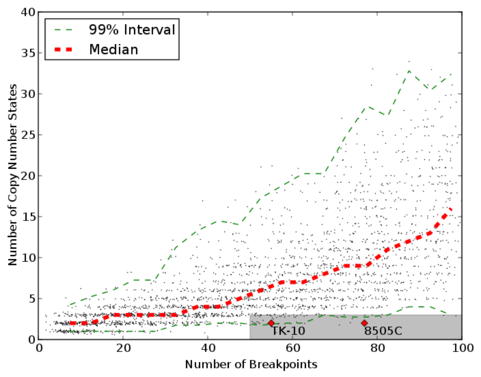


c. Results counting breakpoints as they would appear from paired-end sequencing and counting the number of copy number states needed to cover 95% of the chromosome.

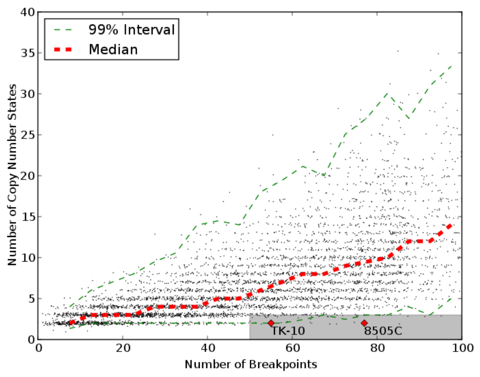


d. Results counting breakpoints as they would appear from microarrays or depth of coverage and counting the number of copy number states needed to cover 95% of the chromosome.

Figure 5.3: Charts of number of breakpoints versus number of copy number states for simulated chromosomes. The shaded gray area indicates the boundaries of the footprint of chromothripsis proposed by Stephens. The red dashed line shows the median number of copy number states for given numbers of breakpoints. The green dashed lines show an interval of copy number states that contains 99% of observations. The Gray region shows the footprint of chromothripsis.



a. Result using paired-end sequencing breakpoint counting.



b. Result using microarray breakpoint counting.

Figure 5.4: Charts of breakpoints versus copy number states for simulations with an overrepresentation of inversions.

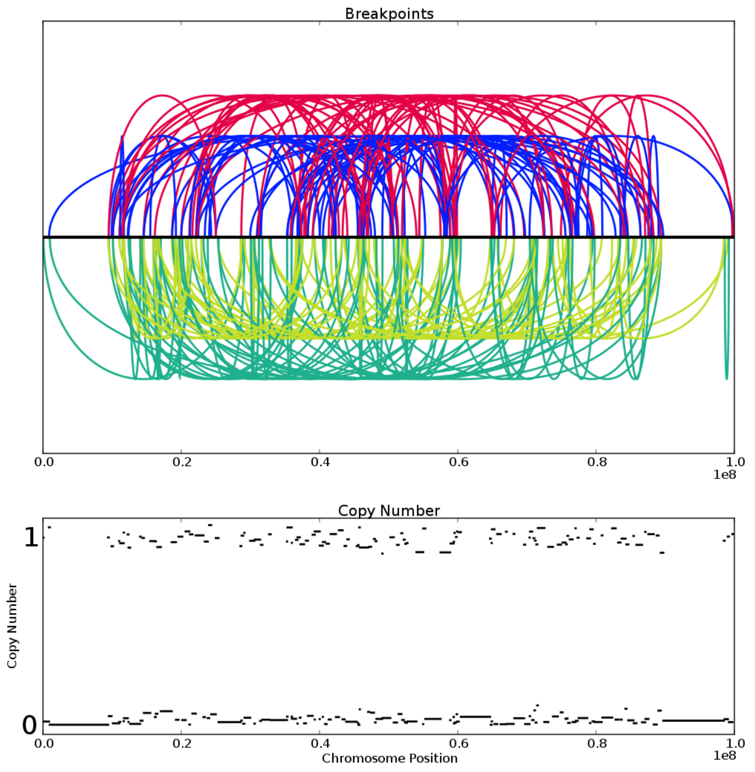


Figure 5.5: Breakpoints and copy numbers of a chromosome simulated with progressive inversions and deletions.

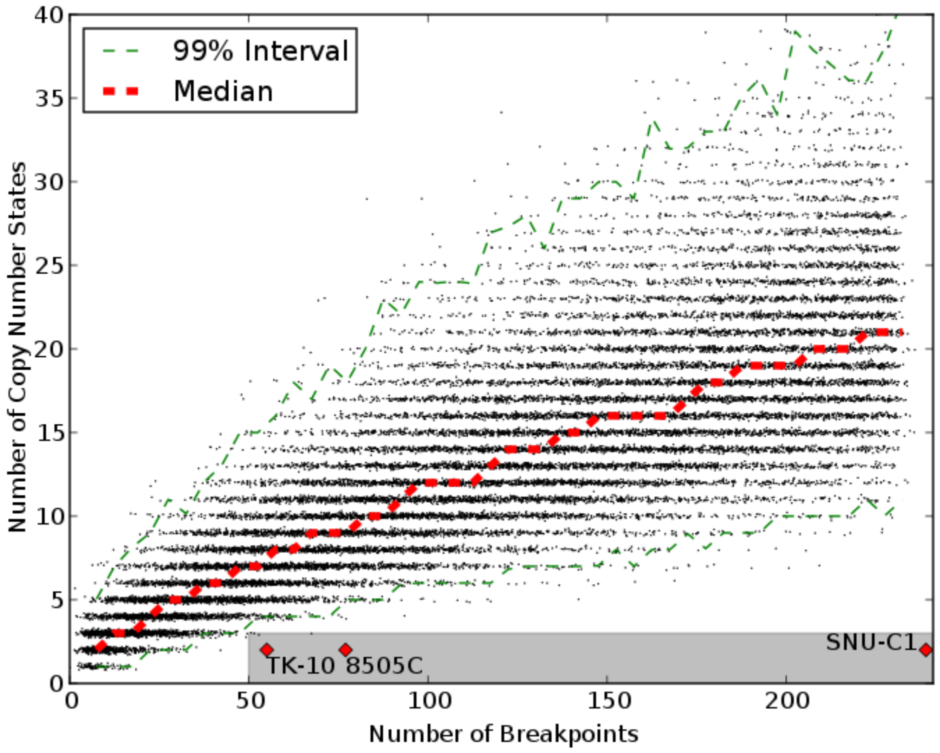


Figure 5.6: Counts of breakpoints and copy number states from a simulation based on the chromosome in Figure 5.5

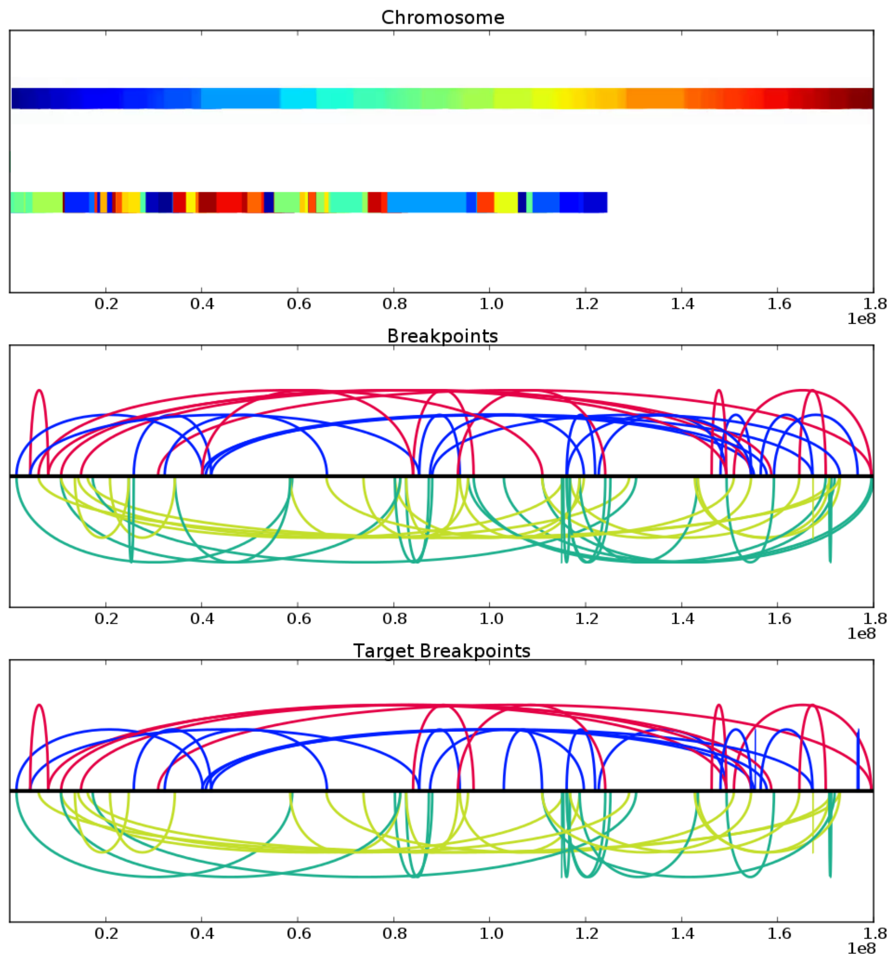


Figure 5.7: An illustration of the result of the series of inversions and deletions for chromosome 5 of TK10. The top panel broadly shows the ordering of segments after re-arrangement. The upper color bar shows all segments of the unrearranged chromosome colored from blue to red. The lower color bar shows segments with the same coloring after rearrangement. Note that some segments have been deleted so the chromosome is shorter. The middle panel shows the breakpoints achieved by inversions and deletions, and the lower panel shows the observed breakpoints.

Appendix A

Supplemental: Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs

Table A.1: Frequency of ambiguously mapping read counts for various read lengths. For each read length, 100,000 fusions were randomly generated. Then for each of these fusions, 200 read pairs spanning the fusion site were generated. The number of read pairs out of these 200 that mapped ambiguously was tabulated. Below is a table of the frequency of ambiguous read pair counts for different read lengths.

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
0	63,997	72,146	77,400	80,247	82,040	86,938	89,202
1	1,733	1,493	435	344	417	140	156
2	1,205	801	738	324	248	114	76
3	1,610	491	364	331	248	88	72
4	958	886	354	224	223	100	51
5	799	619	418	264	161	113	87
6	967	413	246	282	123	121	63
7	745	458	237	194	144	102	64

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
8	624	378	256	185	166	88	84
9	688	328	201	281	140	89	78
10	503	306	251	142	134	83	78
11	526	291	265	146	161	119	80
12	650	262	232	138	161	99	56
13	475	320	197	130	141	77	67
14	385	267	179	119	109	67	74
15	431	239	184	137	131	90	51
16	362	230	165	131	122	81	52
17	347	185	213	127	131	73	54
18	357	202	143	169	101	74	52
19	309	189	137	93	123	83	46
20	227	172	136	115	121	72	52
21	272	188	130	125	111	88	65
22	281	166	112	126	111	72	52
23	212	147	128	153	98	61	69
24	277	172	101	145	89	54	57
25	229	158	159	99	93	66	56
26	202	124	131	133	85	71	61
27	192	134	104	114	67	64	51
28	215	130	117	85	75	46	47
29	155	167	116	86	62	54	55
30	256	116	141	78	58	53	51
31	173	129	124	113	87	57	55
32	144	118	115	104	90	61	63
33	140	122	119	93	75	51	50
34	174	127	111	78	70	58	59

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
35	182	109	101	70	73	56	55
36	163	96	106	88	91	61	38
37	150	101	84	93	97	56	42
38	138	118	64	62	65	50	33
39	146	159	76	79	96	70	36
40	156	120	88	68	99	50	31
41	113	91	85	90	64	68	40
42	132	134	75	73	77	43	47
43	89	95	85	87	72	52	46
44	114	107	103	73	69	45	48
45	117	112	82	70	40	60	54
46	107	87	99	74	50	52	43
47	117	98	110	58	73	31	43
48	106	91	86	67	72	51	32
49	118	73	60	55	64	53	37
50	105	89	78	75	59	61	51
51	100	96	85	75	78	49	55
52	114	75	80	67	77	63	47
53	105	89	69	83	72	65	40
54	83	87	60	54	52	59	40
55	93	86	76	72	56	50	33
56	97	55	81	60	60	70	39
57	124	122	73	61	56	49	37
58	111	89	68	52	48	53	47
59	67	78	61	65	54	40	48
60	83	107	82	65	96	32	39
61	89	68	78	52	86	50	39

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
62	109	85	75	59	68	62	37
63	112	77	63	61	46	42	31
64	85	54	67	69	86	61	33
65	78	61	65	52	47	32	35
66	82	71	51	59	56	46	35
67	94	54	52	74	63	36	32
68	82	72	55	70	69	39	20
69	84	72	55	87	60	31	29
70	100	72	63	56	41	42	27
71	108	56	73	63	55	34	36
72	68	50	58	71	50	42	38
73	68	56	60	65	44	57	33
74	86	62	84	63	58	44	34
75	65	55	51	52	53	48	35
76	90	74	60	55	51	49	30
77	114	75	66	72	45	59	30
78	94	62	72	67	46	43	27
79	72	78	61	75	46	41	27
80	94	75	59	48	49	35	32
81	78	71	67	54	77	41	28
82	61	82	46	53	40	43	26
83	69	74	68	50	65	40	31
84	73	71	63	78	72	31	38
85	71	57	64	59	59	28	21
86	75	59	51	69	58	29	33
87	74	61	43	46	49	60	37
88	57	79	57	55	34	30	37

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
89	67	80	70	54	42	35	22
90	80	66	61	56	57	47	29
91	71	81	77	58	54	40	46
92	68	73	57	45	63	38	31
93	60	59	46	57	56	24	32
94	71	52	68	53	41	25	41
95	77	69	62	63	39	44	33
96	75	82	39	41	44	33	34
97	62	57	53	30	54	35	27
98	58	62	77	40	73	44	36
99	69	58	56	30	54	39	33
100	55	50	54	48	65	38	34
101	70	58	71	55	44	32	28
102	91	57	62	60	55	27	28
103	60	76	50	46	46	23	32
104	56	73	57	35	50	34	46
105	56	64	66	49	47	33	35
106	72	62	45	32	50	31	51
107	55	50	49	47	65	29	51
108	59	60	52	62	61	23	41
109	71	58	51	57	59	16	51
110	69	65	50	67	60	22	36
111	51	59	58	65	61	20	18
112	66	57	57	43	59	33	34
113	64	48	42	64	55	29	25
114	57	49	57	49	43	39	39
115	71	41	49	55	38	41	24

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
116	56	60	67	59	39	33	31
117	50	34	42	60	38	35	33
118	47	59	51	57	40	47	27
119	64	56	44	41	42	32	17
120	56	57	39	57	39	39	26
121	62	57	54	64	44	36	19
122	55	55	68	58	40	44	28
123	45	58	74	65	38	27	21
124	71	62	42	55	51	46	27
125	54	75	67	54	27	49	23
126	59	57	63	45	49	36	23
127	64	45	66	49	47	42	34
128	58	54	58	55	35	34	21
129	58	50	64	51	29	48	23
130	68	60	60	54	25	71	27
131	55	60	39	50	31	51	28
132	59	64	59	25	40	65	36
133	59	74	64	28	51	54	28
134	66	45	51	34	42	58	37
135	61	61	60	52	34	46	29
136	83	65	82	53	33	32	27
137	61	57	53	48	40	28	26
138	88	63	52	50	34	28	18
139	66	64	49	46	55	36	20
140	69	71	59	40	64	24	12
141	71	56	51	24	59	29	22
142	49	47	45	49	49	31	19

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
143	57	53	55	33	48	30	16
144	65	70	51	38	59	32	20
145	59	63	42	52	62	29	27
146	88	80	46	61	47	23	31
147	72	58	47	41	41	17	26
148	36	60	38	37	46	18	16
149	77	71	58	59	46	27	22
150	82	56	57	63	59	21	29
151	69	65	50	62	51	22	23
152	58	50	51	56	62	23	29
153	64	61	55	53	47	25	23
154	73	50	58	67	45	21	21
155	64	74	59	83	93	30	28
156	93	66	56	68	86	29	22
157	65	52	60	63	83	31	26
158	91	77	56	76	72	26	25
159	67	79	57	61	59	28	23
160	70	64	90	97	46	23	19
161	76	55	70	91	39	40	23
162	85	64	96	96	39	29	21
163	49	63	55	75	29	25	24
164	62	75	65	62	38	24	19
165	86	119	100	54	27	22	17
166	85	77	98	48	58	26	20
167	84	85	98	48	38	25	16
168	67	90	94	40	39	24	13
169	78	83	94	45	28	29	15

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
170	106	124	62	26	39	35	11
171	92	124	49	35	40	30	22
172	104	137	72	47	37	23	15
173	105	104	36	49	46	17	16
174	112	99	62	41	36	20	20
175	104	79	42	44	47	29	22
176	124	83	51	63	52	27	19
177	152	70	35	41	38	26	20
178	144	53	48	36	51	29	18
179	126	66	52	38	26	24	16
180	89	50	46	48	33	36	18
181	102	51	49	37	43	32	21
182	101	33	44	39	44	16	26
183	74	47	57	50	60	29	19
184	76	41	42	50	45	31	25
185	77	61	46	31	43	38	28
186	79	48	38	24	36	44	31
187	66	47	37	53	43	28	30
188	60	48	59	63	29	17	59
189	62	62	38	40	32	19	41
190	76	57	33	42	26	31	51
191	62	63	39	50	19	29	57
192	39	54	65	41	28	32	62
193	64	59	71	43	28	38	52
194	78	48	36	41	32	42	54
195	60	57	46	30	46	59	55
196	57	72	63	38	44	92	59

Table A.1 – Continued

Ambiguous Read Count (out of 200)	**Read Length 30**	**Read Length 35**	**Read Length 40**	**Read Length 45**	**Read Length 50**	**Read Length 75**	**Read Length 100**
197	59	75	59	59	76	113	79
198	84	104	145	118	151	136	186
199	254	269	300	336	346	298	421
200	5,673	5,502	5,060	4,809	4,548	3,655	2,856

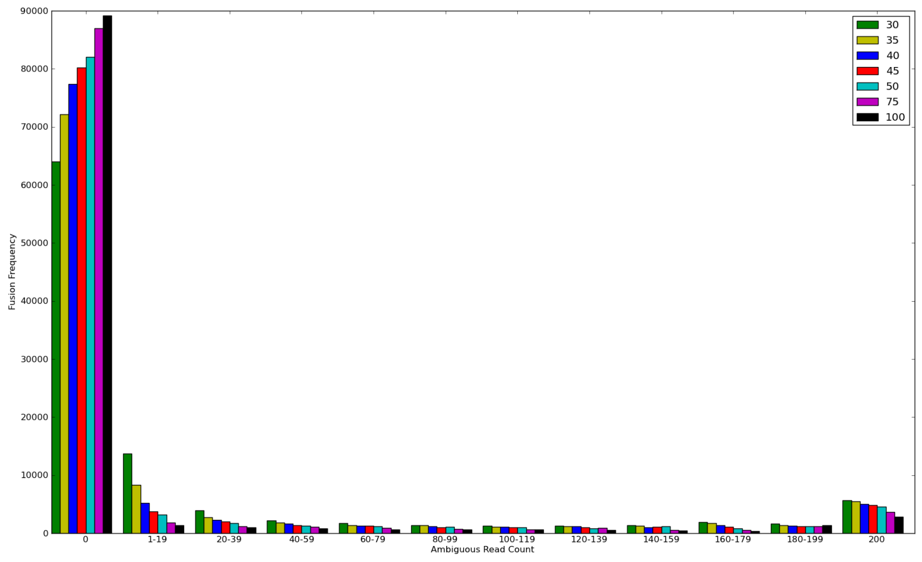


Figure A.1: Graph of ambiguously mapping read count frequency data above.

Table A.2: All gene fusions nominated by discordant read pairs in the simulated data.

Upstream Partner	**Downstream Partner**
FOXO3B	EIF3C
FOXO3B	EIF3CL
FOXO3	EIF3C
FOXO3	EIF3CL

Table A.2 – Continued

Upstream Partner	**Downstream Partner**
FRG1B	LOC162632
FRG1B	LOC220594
FRG1B	USP32
FRG1B	USP6
FRG1	LOC162632
FRG1	LOC220594
FRG1	USP32
FRG1	USP6
LOC283788	LOC162632
LOC283788	LOC220594
LOC283788	USP32
LOC283788	USP6
LOC642236	LOC162632
LOC642236	LOC220594
LOC642236	USP32
LOC642236	USP6
MAGED4B	MBD3L2
MAGED4B	MBD3L3
MAGED4B	MBD3L4
MAGED4B	MBD3L5
MAGED4	MBD3L2
MAGED4	MBD3L3
MAGED4	MBD3L4
MAGED4	MBD3L5
PSG10	PHB
PSG10	ZNF607
PSG11	PHB
PSG11	ZNF607
PSG1	PHB

Table A.2 – Continued

Upstream Partner	**Downstream Partner**
PSG1	ZNF607
PSG2	PHB
PSG2	ZNF607
PSG3	PHB
PSG3	ZNF607
PSG4	PHB
PSG4	ZNF607
PSG5	PHB
PSG5	ZNF607
PSG6	PHB
PSG6	ZNF607
PSG7	PHB
PSG7	ZNF607
PSG8	PHB
PSG8	ZNF607
PSG9	PHB
PSG9	ZNF607
SMN1	CSAG1
SMN1	CSAG2
SMN1	CSAG3
SMN2	CSAG1
SMN2	CSAG2
SMN2	CSAG3

Table A.3: Unambiguous fusion results from melanoma and UHR data. In addition to the ambiguous fusions reported in the results section, our method returned many unambiguous fusions as well, and they are listed below. For the melanoma data, "Previously Reported" indicates whether the fusion was reported by Berger *et al.* Note that the criteria Berger used for reporting was different than that used here. Specifically, Berger required a read to cover the fusion point and excluded read-throughs present in existing databases. For the UHR data, "Previously Reported" indicates whether Maher *et al.* (2009b) reported the fusion. The UHR data used in this study is not the same as used by Maher, but both use sequencing of UHR. We note that all of the fusions reported by Berger are present in our results and nearly all of the read-throughs. The read-throughs we do not report are CDK2-RAB5B, PFKFB4-SCOTIN, FOXRED-TXN2, and C11orf51-C11orf59. The first three were found but excluded because coverage at the fusion site was less than one-twentieth overall coverage. The last was excluded because it lies within an intron of the RefSeq gene LRTOMT.

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
			M000216			
RRM2	2	C2orf48	2	12.0	No	Yes
MFGE8	15	HAPLN3	15	11.0	No	Yes
CLN6	15	CALML4	15	10.0	No	Yes
ARG2	1 4	RAD51L1	14	10.0	No	No
KCTD2	17	ARHGEF12	11	7.0	Yes	No
RPL7	8	EEF1A1	6	6.0	No	No

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
TSC22D4	7	C7orf61	7	6.0	No	Yes
VAX2	2	ATP6V1B1	2	5.0	No	Yes
MGAT5	2	LOC151162	2	4.0	No	Yes
HSPE1	2	MOBKL3	2	4.0	No	Yes
MAGIX	X	PLP2	X	3.0	No	Yes
NDST2	10	NEAT1	11	3.0	No	No
MED20	6	USP49	6	3.0	No	Yes
ARHGEF12	11	C16orf35	16	3.0	No	No
GLT8D4	3	PPP4R2	3	3.0	No	Yes
ZHX1	8	C8orf76	8	3.0	No	Yes
SPP1	4	BRI3BP	12	3.0	No	No
			M000921			
RECK	9	ALX3	1	34.0	Yes	No
HBXIP	1	OR2S2	9	19.0	No	No
C15orf57	15	CBX3	7	10.0	No	No
HERC2	15	MTMR15	15	8.0	No	No
STYXL1	7	TMEM120A	7	6.0	No	Yes
ST6GALNAC6	9	AK1	9	6.0	No	Yes

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
SF3A2	19	AMH	19	5.0	No	Yes
STRADA	17	LIMD2	17	5.0	No	Yes
TIMM23	10	BMS1P4	10	5.0	No	No
APBB3	5	SRA1	5	4.0	No	Yes
BTBD8	1	KIAA1107	1	4.0	No	Yes
TMEM8B	9	TLN1	9	4.0	Yes	No
ASXL1	20	RPL35	9	3.0	No	No
ZNF594	17	FLJ36492	17	3.0	No	No
PIR	X	FIGF	X	3.0	No	Yes
ESRP1	8	DPY19L4	8	3.0	Yes	Yes
PFDN5	12	C12orf10	12	3.0	No	Yes
MTRF1L	6	FBXO5	6	3.0	No	Yes
BBS5	2	KBTBD10	2	3.0	No	Yes
HSPE1	2	MOBKL3	2	3.0	No	Yes
NPL	1	DHX9	1	3.0	No	Yes
GPATCH3	1	GPN2	1	2.0	No	Yes
KIAA1267	17	LRRC37A	17	2.0	No	No
WARS2	1	NOTCH2	1	2.0	No	No

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
VPS45	1	PLEKHO1	1	2.0	No	Yes
MAGIX	X	PLP2	X	2.0	No	Yes
LOC728613	5	SDHA	5	2.0	No	No
TRIM2	4	MND1	4	2.0	No	Yes
			M010403			
SMG5	1	PAQR6	1	12.0	No	Yes
PRMT1	19	C19orf76	19	7.0	No	Yes
RPS4Y1	Y	RPS4X	X	5.0	No	No
STYXL1	7	TMEM120A	7	5.0	No	Yes
SCAMP2	15	WDR72	15	5.0	Yes	No
CLN6	15	CALML4	15	5.0	No	Yes
LOC728190	10	SYT15	10	5.0	No	No
HAVCR1	5	TIMD4	5	5.0	No	Yes
RPS27A	2	UBA52	19	4.0	No	No
SUGT1P	9	NOL6	9	4.0	No	Yes
SMOX	20	LOC728228	20	4.0	No	Yes
ARNTL2	12	C12orf70	12	4.0	No	Yes
RRM2	2	C2orf48	2	4.0	No	Yes

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
ANKRD39	2	ANKRD23	2	3.0	No	Yes
HSPA8	11	RPS11	19	3.0	No	No
LOC541471	2	ANAPC1	2	3.0	No	No
ANXA2	15	RPL6	12	2.0	No	No
ZNF606	19	C19orf18	19	2.0	No	Yes
NONO	X	RPL6	12	2.0	No	No
UNQ2963	12	CLSTN3	12	2.0	No	Yes
ABCB8	7	ACCN3	7	2.0	No	Yes
HOXD11	2	HOXD10	2	2.0	No	Yes
UBXN2A	2	MFSD2B	2	2.0	No	Yes
POLA2	11	CDC42EP2	11	2.0	No	Yes
			M970109			
UCN2	3	PFKFB4	3	57.0	No	Yes
HNRNPU	1	NCRNA00201	1	15.0	No	Yes
CLR	12	CLEC2D	12	9.0	No	Yes
BTBD8	1	KIAA1107	1	7.0	No	Yes
SLC39A1	1	CRTC2	1	6.0	No	Yes
RASSF8	12	SSPN	12	5.0	No	No

Table A.3 – Continued

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
			M980409			
CDK2	12	RAB5B	12	30.0	Yes	Yes
UCN2	3	PFKFB4	3	24.0	No	Yes
TLK2	17	LOC100288069	1	15.0	No	No
CLTC	17	TMEM49	17	14.0	Yes	Yes
SLC39A1	1	CRTC2	1	13.0	No	Yes
ABCB8	7	ACCN3	7	9.0	No	Yes
ARPC4	3	TTLL3	3	8.0	No	Yes
ARL6IP1	16	RPS15A	16	7.0	No	Yes
GCN1L1	12	PLA2G1B	12	7.0	Yes	No
SDHAF2	11	C11orf66	11	6.0	No	Yes
STYXL1	7	TMEM120A	7	6.0	No	Yes
MRPS10	6	GUCA1B	6	5.0	No	Yes
OTUD6B	8	LRRC69	8	4.0	No	Yes
LOC728613	5	SDHA	5	4.0	No	No
POLA2	11	CDC42EP2	11	3.0	No	Yes
ADSL	22	SGSM3	22	3.0	No	Yes
CCDC15	11	SLC37A2	11	3.0	Yes	Yes

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
CLR	12	CLEC2D	12	3.0	No	Yes
ANKRD39	2	ANKRD23	2	3.0	No	Yes
			M990802			
ANKHD1	5	C5orf32	5	65.0	Yes	No
RB1	13	ITM2B	13	26.0	Yes	No
SMG5	1	PAQR6	1	12.0	No	Yes
LRRFIP1	2	RBM44	2	6.0	No	Yes
TPD52L2	20	DNAJC5	20	5.0	No	Yes
OR51B4	11	HBE1	11	4.0	No	Yes
WRB	21	SH3BGR	21	3.0	No	No
KIAA1467	12	EMP1	12	3.0	No	No
RPL11	1	TCEB3	1	3.0	No	Yes
YARS2	12	NAP1L1	12	3.0	No	No
TTLL12	22	EIF1	17	3.0	No	No
NFX1	9	MTRNR2L8	11	2.0	No	No
			M980928			
HOXD4	2	HOXD3	2	9.0	No	Yes
NAIP	5	OCLN	5	6.0	No	No

Table A.3 – Continued

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
SLAMF9	1	IGSF9	1	2.0	No	Yes
			M990514			
UCN2	3	PFKFB4	3	19.0	No	Yes
NADSYN1	11	LOC100188947	10	17.0	No	No
PRMT1	19	C19orf76	19	16.0	No	Yes
COL7A1	3	UCN2	3	15.0	No	Yes
PACSIN2	22	ARFGAP3	22	13.0	No	Yes
CECR7	22	IL17RA	22	10.0	No	Yes
XRCC1	19	ETHE1	19	10.0	No	No
SLC39A1	1	CRTC2	1	9.0	No	Yes
GALNT8	12	KCNA6	12	8.0	No	Yes
WDR35	2	TTC32	2	8.0	Yes	Yes
C14orf133	14	C14orf148	14	7.0	No	Yes
MRPL20	1	CCNL2	1	6.0	No	Yes
C7orf50	7	COX19	7	6.0	No	No
PTPRG	3	C3orf14	3	6.0	Yes	Yes
DMPK	19	SIX5	19	6.0	No	Yes
SLC29A1	6	HSP90AB1	6	6.0	No	Yes

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
GPR153	1	ICMT	1	6.0	Yes	No
C7orf68	7	EFCAB3	17	5.0	No	No
PIM2	X	SLC35A2	X	5.0	No	Yes
USP36	17	CYTH1	17	5.0	No	Yes
NDUFS2	1	FCER1G	1	5.0	No	Yes
UBE4A	11	ATP5L	11	5.0	No	Yes
STYXL1	7	TMEM120A	7	4.0	No	Yes
PKD1	16	NPIP	16	4.0	No	No
CADM4	19	ZNF428	19	4.0	No	Yes
SHC1	1	PYGO2	1	4.0	No	Yes
CD151	11	TSPAN4	11	4.0	Yes	Yes
KDM6B	17	TMEM88	17	4.0	No	Yes
BCL2L2	14	PABPN1	14	4.0	No	Yes
TPD52L2	20	DNAJC5	20	4.0	No	Yes
ST6GALNAC6	9	AK1	9	4.0	No	Yes
WRB	21	SH3BGR	21	4.0	No	No
SIRT7	17	PCYT2	17	4.0	No	Yes
ODF3B	22	TYMP	22	4.0	No	Yes

Table A.3 – Continued

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
SNX8	7	FTSJ2	7	3.0	No	No
POLA2	11	CDC42EP2	11	3.0	No	Yes
PGAP1	2	C2orf66	2	3.0	No	Yes
LMCD1	3	NAG-7	3	3.0	No	Yes
TSEN34	19	RPS9	19	3.0	No	No
CLR	12	CLEC2D	12	3.0	No	No
SIDT2	11	TAGLN	11	3.0	No	Yes
C8orf58	8	KIAA1967	8	3.0	No	Yes
UBE2J2	1	FAM132A	1	2.0	No	Yes
CUEDC1	17	MRPS23	17	2.0	No	Yes
C1RL	12	C1R	12	2.0	No	Yes
GPR155	2	CIR1	2	2.0	No	No
			501_Mel			
CCT3	1	C1orf61	1	84.0	Yes	No
SLC12A7	5	C11orf67	11	77.0	Yes	No
GNA12	7	SHANK2	11	36.0	Yes	No
FCHSD2	11	P2RY6	11	21.0	No	No
CLN6	15	CALML4	15	5.0	No	Yes

Table A.3 – Continued

5’ Gene	**Chromosome**	**3’ Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
ANP32B	9	ATP5I	4	5.0	No	No
RAB6A	11	EYS	6	4.0	No	No
MANBA	4	UBE2D3	4	4.0	No	No
RBBP5	1	NUAK2	1	4.0	No	No
DUS3L	19	PRR22	19	3.0	No	Yes
C15orf57	15	CBX3	7	3.0	No	No
FRG1	4	GOSR1	17	3.0	No	No
TBCEL	11	TECTA	11	2.0	No	Yes
CLR	12	CLEC2D	12	2.0	No	Yes
PARP1	1	MIXL1	1	2.0	Yes	No
			MeWo			
ARL6IP1	16	RPS15A	16	10.0	No	Yes
METTL10	10	FAM53B	10	7.0	No	Yes
ZNF654	3	C3orf38	3	5.0	No	Yes
TRAK2	2	ALS2CR12	2	4.0	No	Yes
UBA2	19	WTIP	19	4.0	No	Yes
STYXL1	7	TMEM120A	7	3.0	No	Yes
FBXL19	16	ORAI3	16	3.0	No	Yes

Table A.3 – Continued

5’ Gene	Chromosome	3’ Gene	Chromosome	Supporting Read Pairs	Previously Reported	Read-through
CORO1C	12	SELPLG	12	2.0	No	Yes
WRN	8	FIBP	11	2.0	No	No
SEMA4C	2	ANKRD39	2	2.0	No	Yes
HARS	5	DND1	5	2.0	No	Yes
C20orf29	20	MAVS	20	2.0	No	Yes
UHR						
BCAS4	20	BCAS3	17	75.0	Yes	No
GAS6	13	RASA3	13	33.0	Yes	No
AP3D1	19	JSRP1	19	16.0	No	No
ARFGEF2	20	SULF2	20	13.0	Yes	No
CLN6	15	CALML4	15	11.0	No	Yes
RPS6KB1	17	TMEM49	17	8.0	Yes	No
B3GAT3	11	GANAB	11	7.0	No	Yes
BCR	22	ABL1	9	7.0	Yes	No
RRM2	2	C2orf48	2	7.0	No	Yes
ADCK4	19	NUMBL	19	6.0	Yes	Yes
RPLP0	12	EEF1AL7	4	6.0	No	No
SYTL2	11	PICALM	11	6.0	No	No

Table A.3 – Continued

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
RPL3	22	EEF1AL7	4	5.0	No	No
SIDT2	11	TAGLN	11	5.0	No	Yes
HNRNPUL2	11	C11orf49	11	5.0	No	No
GCN1L1	12	MSI1	12	5.0	No	No
NUP214	9	XKR3	22	5.0	Yes	No
CCDC123	19	PEPD	19	4.8	Yes	No
TANC2	17	CA4	17	4.0	No	No
EEF1AL7	4	GAPDH	12	4.0	No	No
ZFP41	8	GLI4	8	4.0	Yes	Yes
VAMP8	2	VAMP5	2	4.0	Yes	Yes
HNRNPU	1	NCRNA00201	1	4.0	No	Yes
EEF1AL7	4	EEF2	19	4.0	No	No
LIME1	20	SLC2A4RG	20	4.0	No	Yes
SAPS3	11	DPP3	11	4.0	No	No
SSSCA1	11	FAM89B	11	4.0	No	Yes
VPS72	1	TMOD4	1	3.0	No	Yes
SMG5	1	PAQR6	1	3.0	No	Yes
ANKRD39	2	ANKRD23	2	3.0	Yes	Yes

Table A.3 – Continued

5' Gene	**Chromosome**	**3' Gene**	**Chromosome**	**Supporting Read Pairs**	**Previously Reported**	**Read-through**
RPS10	6	NUDT3	6	3.0	No	Yes
POLA2	11	CDC42EP2	11	3.0	Yes	Yes
MRPS10	6	GUCA1B	6	3.0	No	Yes
SUGT1P	9	NOL6	9	3.0	No	Yes
CTNNBIP1	1	CLSTN1	1	3.0	No	Yes
VIM	10	EEF1AL7	4	3.0	No	No
AHCYL1	1	CFL1	11	2.0	No	No
CBFA2T3	16	LOC390748	16	2.0	No	Yes
TAGLN2	1	CCDC19	1	2.0	No	Yes
FKBP4	12	ITFG2	12	2.0	No	Yes
EEF2	19	EEF1AL7	4	2.0	No	No
EEF1AL7	4	SND1	7	2.0	No	No
BAIAP2L2	22	SLC16A8	22	2.0	No	Yes
LOC442454	X	ENO1	1	2.0	No	No
ALKBH6	19	C19orf46	19	2.0	No	Yes
PHRF1	11	DRD4	11	2.0	No	No
SPN	16	QPRT	16	2.0	No	Yes

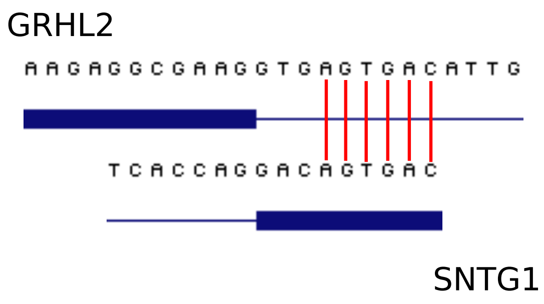


Figure A.2: A short homologous sequence near the fusion site of GRHL2 and SNTG1.

A.1 Ambiguous fusion sequences.

Below is the sequence surrounding each fusion site for the ambiguous fusions reported in the results.

A.1.1 HOMEZ-MYH6

CTCGGAGCGGCCGCACCGGGCAGCAACCCCACTCCCACTCGGAGGCCCCCTGCCCTCTCCCC CACTTCCCCCCGGCCCATGGTGCGAGGCTGGGAGCCGCCGCCCGGGCTGGACTGC *GGCCTCCCT GGAGCACGAGGAGGGCAAGATCCTCCGGGCCCAGCTAGAGTTCAACCAGA TCAAGGCAGAGATCGAGCGGAAGCTGGCAGAGAAGGACGAGGAGATGGA ACAGGCCAAGCGCAACCACCAGCGGGTGGTGGACTCGCTGCAGACCTCCC TGGATGCAGAGACACGCAGCCGCAACGAGGTCCTGAG*

A.1.2 KIAA1267-ARL17A

GGGGAGTCTGATATTGAAGAGGAAGAACTGACCAGAGCTGATCCCGAGCAGCGTCATGTACC CCTGAGACGCAGGTCAGAATGGAAATGGGCTGCAGACCGGGCAGCTATTGTCAGCCGCTGGAACTGG CTTCAGGCTCATGTTTCTGACTTGGAATATCGAATTCGTCAGCAAACAGACATTTACAAACAGATAC GTGCTAATAAG *GTTTCTGTGTGGAGACAGTAGAATATAAAAATAACACCTTCG CTGTCTGGGATGTTGGCAGCCACTTCAAAATCAGACCTCTGTGGCAGCATT TTTTCCAGAACACAAAAGGTGCCAGAAGCCCAGGAAGCACACATCAAGGC TCACTTGCCAGCGGGGTGCTGCCAATAAAATGTAGTCACGTGGAATTTGGA*

ATGTGG
Note that this fusion sequence matches GenBank mRNA accession BC006271.1.

A.1.3 CPEB1-RPS17

GCCGACAGTAACTTTGTCCGGAGCCCATCTCAGAGGCTTGACCCCAGCAGGACGGTGTTTGT
CGGTGCTCTGCATGGAATGCTAAATGCTGAGGCCCTGGCAGCCATCTTGAACGACCTATTTGGTGGA
GTGGTGTATGCCGGGATTGACACAGATAAGCACAAGTATCCCATTGGTTCTGGTCGTGTGACTTTCA
ATAACCAACGGAGTTACCTGAAAGCAGTCAGCGCTGCTTTTGTGGAGATCAAAACCACCAAGTTCAC
AAAGAAGGTTCAGATTGACCCCTACCTAGAAGATTCTCTGTGTCATATCTGCAGTTCTCAGCCTGGT
CCTTTCTTCTGTCGAGATCAG *GTTTCCTCTTTTACCAAGGACCCGCCAACATGGGC*
CGCGTTCGCACCAAAACCGTGAAGAAGGCGGCCCGGGTCATCATAGAAAA
GTACTACACGCGCCTGGGCAACGACTTCCACACGAACAAGCGCGTGTGCG
AGGAGATCGCCATTATCCCCAGCAAAAAGCTCCGCAACAAGATAGCAGGT
TATGTCACGCATCTGATGAAGCGAATTCAGAGAGGCCCAGTAAGAGGTAT
CTCCATCAAGCTGCAGGAGGAGGAGAGAGAAAGGAGAGACAATTATGTTC
CTGAGGTCTCAGCCTTGGATCAGGAGATTATTGAAGTAGATCCTGACACTA
AGGAAATGCTGAAGCTTTTGGACTTCGGCAGTCTGTCCAACCTTCAGGTCA
CTCAGCCTACAGTTGGGATGAATTTCAAAACGCCTCGGGGACCTGTTTGAA
TTTTTTCTGTAGTGCTGTATTATTTTCAATAAATCTGGG

A.1.4 PPIP5K1-CATSPER2

GTCCAGGAAAGGCATCAGATGAACCAGACCGGGCATTGCAGACTTCACCCCAGCCTCCTGA
GGGCCCTGGCCTTCCGAGGAGATCACCCCTCATTCGTAACCGAAAAGCTGGTTCCATGGAGGTACTT
TCTGAGACTTCATCCTCGAGGCCTGGTGGCTACCGGCTCTTTTCATCTTCACGGCCACCCACAGAAA
TGAAGCAGAGTGGCCTAG *ATCCTTCCCGCCAGAAGAAACTTGTATTGGGAGATCA*
ACACCAGCTAGTGCGTTTCTCTATAAAGCCTCAGCGTATAGAACAGATTTC
ACATGCCCAGAGGCTGTTGAGCAGGCTTCATGTGCGCTGCAGTCAGAGGC
CACCT

Appendix B

Supplemental: Combinatorics of the Breakage-Fusion-Bridge Mechanism

B.1 Proofs

Theorem 3: Algorithm CheckBFB checks if x is a BFB-string in $O(n)$ time.

Proof. (*Maximal Palindromes*) A string is a BFB-string if and only if it can be formed by an inverted prefix duplication from another BFB-string or it is the original string, x^0. An inverted prefix duplication forms an even palindrome at the beginning of a string. CheckBFB finds such a palindrome and then, in effect, recurses on the string that ends at that palindrome's center. The Suffix Lemma guarantees that that string must be a BFB-string if x is a BFB-string. If a string does not begin with a palindrome, and it is not x^0, then it is not a BFB-string.

We would like to describe an algorithm that finds the radius of the largest palindrome centered at each interstice in a string in linear time. We will, in fact, describe two. The first lends itself to clear exposition and establishes that the problem can be solved in linear time. The second lends itself to easier implementation and is the method we used for results in this paper.

Our first algorithm has three steps and is similar to the algorithm described by Gusfield [24].

1. Build a generalized suffix tree from the string and the reverse of the string, that

is, a suffix tree that contains both suffixes and prefixes of the string. As usual, each leaf corresponding to a suffix beginning at position i is labeled i. Each leaf corresponding to a prefix ending at i is labeled $-i$. It is well known this can be done in linear time.

2. For each position in the string, i, find the lowest common ancestor of i and $-i$. Using Tarjan's algorithm, this can be done with linear preprocessing time and constant query time [27].
3. Find the distance from the root to the LCA. This is the size of the largest common prefix ending and i and suffix starting at i and hence radius of the largest palindrome.

While this is indeed a linear time algorithm to find maximal palindromes, Tarjan's algorithm is not considered to be particularly implementable. So, we implemented another linear time algorithm that is similar to Manacher's algorithm [50]. This algorithm proceeds through each interstice in the string, from left to right. As each interstice, is visited, pairs of characters at the boundary of the current known palindrome are tested to see if that palindrome can be expanded. If it cannot, then the palindrome radius is appended to the R array. Then, for each interstice to the right lying in the radius of this palindrome, the corresponding maximal palindrome size for the interstice lying to the left is checked. If the maximal palindrome to the left lies completely within the radius of the current palindrome, then we know that the radius of the maximal palindrome to the right is equal to the radius of the maximal palindrome on the left. If the maximal palindrome on the left is a proper prefix of the current palindrome or it extends beyond the radius of the current palindrome, then the corresponding palindrome on the right has a radius that extends at least to the current palindrome's radius.

Algorithm *MaximalPalindromes*

Input: String s

Output: Array of integers R, the maximal palindrome radii

```
1.  $i = 0$
2.  $\ell = 0$
3.  $n = $ len(s)
4.  **while** $i + \ell < n - 1$
```

```
5.      do if s[$i$-$\ell$] = s[$i$+$\ell$+1]
6.          then $\ell$++
7.              continue
8.      $R$.append($\ell$)
9.      $c$ = 0
10.     $foundSuffix$ = False
11.     for $0 \leq i' < \ell$ - 1
12.         do if $\ell - i' - 1 \leq R(i - i' - 1)$
13.             then $\ell$ = $\ell - i' - 1$
14.                 $foundSuffix$ = True
15.                 break
16.         $R$.append(min($\ell - i' - 1$,$R(i - i' - 1)$))
17.         $c$++
18.     if not $foundSuffix$
19.         then $\ell$ = 0
20.     $i$ += $c$ + 1
21. return $R$
```

Note that if the first if clause is true, then ℓ is incremented by one. If the inner loop iterates all ℓ times, then ℓ is set to zero and i is increased by $\ell+1$. If the inner loop finds a palindromic suffix and breaks, then ℓ is set to $\ell - i' - 1$ and i is set to $i + i' + 2$. In all three cases, $i+\ell$ is incremented by one in each iteration of the outer loop. Therefore the outer loop only iterates n times. The inner loop appends a value to R in iteration and thus is called at most $n-1$ times. Thus, $MaximalPalindromes$ is in $O(n)$.

Finally, note that both algorithms presented above give us the radius of largest palindrome measured from the center of the palindrome. The algorithm CheckBFB uses the length of largest palindrome measured from its leftmost character to its center, so we need to convert the R array we get from MaximalPalindromes to the P array used by *CheckBFB*. This can easily be done in linear time:

Algorithm *RtoP*
Input: R array (palindrome radii)
Output: P array (palindrome lengths from left)

```
1.  (∗ Instantiate P array to all zeroes ∗)
2.  P = [0]*(R.length() + 1)
3.  for $0 < i \leq R.length()$
4.      do P[i - R[i] + 1] = max(P[i - R[i] + 1], R[i])
5.  pal_counter = 0
6.  for $0 < i \leq P.length()$
7.      do pal_counter = max(P[i], pal_counter - 1, 0)
8.         P[i] = max(P[i], pal_counter)
9.  return P
```

RtoP reads through the *R* array once. Each time it finds a nonzero palindrome radius, it updates the appropriate element in the *P* array. Then, it reads through the *P* array once and updates each value so it is the maximum of its current value or the appropriately decremented value for a palindrome that contains the element. □

B.2 Applying BFB Rules

Some of the rules are “reductions" rather than simple tests if a count-vector admits a BFB schedule. This means that they show that a count-vector admits a BFB schedule if (and sometimes only if) some set of simpler count-vectors all admit a BFB schedule. We therefore apply these rules in a recursive fashion outlined below:

Algorithm *ApplyRules*
Input: $\vec{n}$ a count-vector
Output: **true** if $\vec{n}$ admits a BFB schedule, **false** if $\vec{n}$ does not admit a BFB schedule, or *R* a set of count-vectors such that $\vec{n}$ admits a BFB schedule iff every count-vector in *R* does

```
1.  if $\vec{n}$ passes $CountThresholdRule$ return true
2.  if $\vec{n}$ fails $RuleOfOne$, $OddEvenRule$, or $DivByFourRule$ return false
3.  $\vec{o} = OddReduction(\vec{n})$
4.  if $\vec{o}.length() = \vec{n}.length()$
5.        (∗ If the whole count-vector is odd ∗)
```

```
6.      then return ApplyRules($\vec{o}$)
7.      else  if not ApplyRules($\vec{o}$)
8.              then return false
9.  $\vec{f} = FourReduction(\vec{n})$
10. if not ApplyRules($\vec{f}$) return false
11. $\vec{p}, \vec{s} = TwoReduction(\vec{n})$
12. if $\vec{p}$ or $\vec{s}$ not empty
13.     then if not ApplyRules($\vec{p}$) or not ApplyRules($\vec{s}$) return false
14.             if ApplyRules($\vec{p}$) and ApplyRules($\vec{s}$) return true
15.     else  return ApplyRules($\vec{p}$) $\cup$ ApplyRules($\vec{s}$)
16. return $\vec{n}$
```

B.3 Analysis of BFB_Tree

Let $I(v)$ denote the multiplicity of a node v that is independent, and 0 otherwise. At step j of the algorithm, we must assign the n_j nodes at the current level to the independent nodes at level $j-1$. Let n_{jv} be the nodes assigned to node v. Then, all assignments must satisfy $\sum_v I(v) n_{jv} = n_j$. We start with $n_k = 1$, and continue to the first level where the number of nodes is more than one. Upto this point, there is a unique assignment. For all subsequent levels j, the number of independent nodes at level j is at most $\lceil \frac{n_j}{2} \rceil$, and except for the center node, each has a multiplicity of at least 2. The total number of assignments at the level is bounded by

$$\binom{\lceil \frac{n_j}{2} \rceil + \lfloor \frac{n_{j-1}}{2} \rfloor - 1}{\lfloor \frac{n_{j-1}}{2} \rfloor} \leq 2^{\frac{1}{2}(n_{j-1}+n_j)}$$

And the total number of trees considered is bounded by

$$\prod_j 2^{\frac{1}{2}(n_{j-1}+n_j)} \leq 2^{\sum_j n_j} = 2^n$$

By definition, these assignments will satisfy mirror-symmetry, and the long-end property is easily checked as well. However, checking all pair-symmetry constraints at level j requires a single traversal of the tree.

However, the actual performance should be superior. The algorithm also requires maintaining the count-vector I. Note that $I(r) = 1$ for the root node r. For other nodes v with children $v_{-\ell}, \ldots, v_{\ell}$

$$I(v_j) = \begin{cases} 2I(v) & -\ell \leq j < 0 \\ I(v) & j = 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that once a node is dependent ($I(v) = 0$), its descendants remain dependent. Further, the multiplicity of the node is a power of 2, and is doubled each time an independent node is a non-central child of its parent. As the multiplicities increase, the number of assignments decreases quickly, so that in practice, there will be very few valid assignments after the first few levels.

Specifically, consider an input data-set with $k+1$ levels, a single node at level $k+1$, and a maximum of m nodes at each of the k levels.

let I_t be the multiplicity of the left-most node at level t from a random assignment. The long-ends property ensures that $I_t \neq 0$, Then,

$$I_t = \begin{cases} I_{t+1} & \text{(* v is the only child of its parent*)} \\ 2I_{t+1} & \text{(* otherwise*)} \end{cases}$$

It is not hard to show that I_t almost doubles with each round, and reaches the maximum value m after $\log_2(m)$ rounds. Once $I_t = n$, for some t there is only one assignment possible. Prior to that, the maximum number of assignments at each level is $O(2^m)$. Therefore, the number of labeled trees with long-ends and mirror symmetry is bounded by $O(2^{m \log_2 m})$.

Appendix C

Supplemental:An algorithmic approach for breakage-fusion-bridge detection in tumor genomes

C.1 Properties of BFB Strings

In this section, we prove Claim 1 from Chapter 4. To do so, we first formulate several auxiliary claims.

Observation 1. *If $\alpha \xrightarrow{BFB} \beta$, $\beta = \beta'\beta''$, and $\beta'' \xrightarrow{BFB} \beta''\gamma$, then $\alpha \xrightarrow{BFB} \beta\gamma$.*

Call a string α an *l-t-string* if for the count vector $\vec{n}(\alpha) = [n_1, n_2, \ldots, n_k]$, $n_r > 0$ if and only if $l \leq r \leq t$. Thus, an l-t-BFB string is an l-BFB string α such that $top(\alpha) = t$. Denote by $\alpha_{l,t}$ the consecutive genomic region $\alpha_{l,t} = \sigma_l \sigma_{l+1} \ldots \sigma_t$ (when $t < l$, $\alpha_{l,t} = \varepsilon$), and observe that l-t-BFB strings always start with the prefix $\alpha_{l,t}$.

Claim 5. *Let l', l, t be integers and $\alpha\beta$ an l'-BFB string such that β is an l-t-string. Then,*

1. *If β starts with the prefix $\alpha_{l,t}$, then $\alpha_{l,t} \xrightarrow{BFB} \beta$ (i.e. β is an l-t-BFB string).*
2. *If β ends with the suffix $\alpha_{l,t}$, then $\bar{\alpha}_{l,t} \xrightarrow{BFB} \bar{\beta}$.*
3. *If β starts with the prefix $\bar{\alpha}_{l,t}$, then $\bar{\alpha}_{l,t} \xrightarrow{BFB} \beta$.*

4. *If β ends with the suffix $\bar{\alpha}_{l,t}$, then $\alpha_{l,t} \xrightarrow{BFB} \bar{\beta}$.*

Proof. When $t < l$, $\alpha_{l,t} = \beta = \varepsilon$, and all four items in the claim are sustained in a straightforward manner. Similarly, when $\alpha\beta = \alpha_{l',t}$, then $\beta = \alpha_{l,t}$ and again all four items in the claim are sustained. Otherwise, $t \geq l$ and there are some ρ, γ such that $\gamma \neq \varepsilon$, $\rho\gamma$ is an l'-BFB string, and $\alpha\beta = \rho\gamma\bar{\gamma}$. In particular, $\alpha_{l',t}$ is a proper prefix of $\alpha\beta$. Assume by induction that the claim is sustained with respect to all proper prefixes of $\alpha\beta$ (from Lemma 2 in [36], all such prefixes are l'-BFB strings). Note that β, $\bar{\gamma}$, and $\gamma\bar{\gamma}$ are all suffixes of $\alpha\beta = \rho\gamma\bar{\gamma}$. Consider three cases: 1. β is a proper suffix of $\bar{\gamma}$, 2. β is a proper suffix of $\gamma\bar{\gamma}$ and $\bar{\gamma}$ is a suffix of β, and 3. $\gamma\bar{\gamma}$ is a suffix of β.

1. β is a proper suffix of $\bar{\gamma}$. In this case, $\bar{\gamma} = \gamma'\beta$ for some string $\gamma' \neq \varepsilon$, therefore $\rho\gamma\bar{\gamma} = \rho\bar{\beta}\bar{\gamma}'\gamma'\beta$. From the inductive assumption and the fact that $\rho\bar{\beta}$ is a proper prefix of $\rho\bar{\beta}\bar{\gamma}' = \rho\gamma$ (which is in turn a proper prefix of $\alpha\beta$), $\rho\bar{\beta}$ sustains the claim. Therefore,

1. If β starts with the prefix $\alpha_{l,t}$, then $\bar{\beta}$ ends with the suffix $\bar{\alpha}_{l,t}$, therefore $\alpha_{l,t} \xrightarrow{\text{BFB}} \beta$.

2. If β ends with the suffix $\alpha_{l,t}$, then $\bar{\beta}$ starts with the prefix $\bar{\alpha}_{l,t}$, therefore $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\beta}$.

3. If β starts with the prefix $\bar{\alpha}_{l,t}$, then $\bar{\beta}$ ends with the suffix $\alpha_{l,t}$, therefore $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \beta$.

4. If β ends with the suffix $\bar{\alpha}_{l,t}$, then $\bar{\beta}$ starts with the prefix $\alpha_{l,t}$, therefore $\alpha_{l,t} \xrightarrow{\text{BFB}} \bar{\beta}$.

2. β is a proper suffix of $\gamma\bar{\gamma}$ and $\bar{\gamma}$ is a suffix of β. In this case, there are some γ_1 and γ_2 such that $\gamma_1 \neq \varepsilon$, $\gamma = \gamma_1\gamma_2$, $\gamma\bar{\gamma} = \gamma_1\gamma_2\bar{\gamma}_2\bar{\gamma}_1$ and $\beta = \gamma_2\bar{\gamma}_2\bar{\gamma}_1$. Thus, $\alpha\beta = \rho\gamma\bar{\gamma} = \rho\gamma_1\gamma_2\bar{\gamma}_2\bar{\gamma}_1 = \rho\bar{\beta}\bar{\gamma}_1$. Here also, we get that $\rho\bar{\beta}$ is a proper prefix of $\alpha\beta$, and similarly as in the previous case the inductive assumption implies the correctness of the claim.

3. $\gamma\bar{\gamma}$ is a suffix of β. In this case, there is some γ' such that $\beta = \gamma'\gamma\bar{\gamma}$, and therefore $\alpha\beta = \alpha\gamma'\gamma\bar{\gamma}$. To show items (1) and (3) in the claim, assume that β starts with the prefix ϕ such that either $\phi = \alpha_{l,t}$ or $\phi = \bar{\alpha}_{l,t}$, respectively. It must be that ϕ is a prefix of $\gamma'\gamma$, since the first character of $\bar{\gamma}$ is the reverse of the last character of γ, and thus cannot be included in ϕ. Therefore, from the inductive assumption and the fact that $\alpha\gamma'\gamma$ is a proper prefix of $\alpha\gamma'\gamma\bar{\gamma} = \alpha\beta$ (recall that $\gamma \neq \varepsilon$ and therefore $\bar{\gamma} \neq \varepsilon$), $\phi \xrightarrow{\text{BFB}} \gamma'\gamma$. By definition, $\phi \xrightarrow{\text{BFB}} \gamma'\gamma\bar{\gamma} = \beta$, proving items (1) and (3) in the claim.

To show items (2) and (4) in the claim, assume that β ends with the suffix ϕ such that either $\phi = \alpha_{l,t}$ or $\phi = \bar{\alpha}_{l,t}$, respectively. Similarly as above, it must be that ϕ is a suffix of $\bar{\gamma}$. Note that case (2) of this proof implies that $\bar{\phi} \xrightarrow{\text{BFB}} \gamma$, and by definition $\bar{\phi} \xrightarrow{\text{BFB}} \gamma\bar{\gamma}$. In particular, $\bar{\phi}$ is a prefix of γ, and therefore the string $\alpha\gamma'\bar{\phi}$ is a proper prefix of $\alpha\beta = \alpha\gamma'\gamma\bar{\gamma}$, and $\bar{\phi}$ is the suffix of the suffix $\gamma'\bar{\phi}$ of $\alpha\gamma'\bar{\phi}$. From the inductive assumption, $\phi \xrightarrow{\text{BFB}} \phi\bar{\gamma'}$. Thus, from Observation 1, and the fact that ϕ is a suffix of $\bar{\gamma}$, we get that $\bar{\phi} \xrightarrow{\text{BFB}} \gamma\bar{\gamma} \xrightarrow{\text{BFB}} \gamma\bar{\gamma}\bar{\gamma'} = \bar{\beta}$, and items (2) and (4) in the clam follow. □

Claim 6. *Let α be a BFB string, and let $\sigma\beta\bar{\sigma}$ be a substring of α such that β contains no occurrences of σ or $\bar{\sigma}$. Then, β is a palindrome.*

Proof. From Lemma 2 in [36], every prefix of α is a BFB string, and thus we may assume without loss of generality that $\sigma\beta\bar{\sigma}$ is a suffix of α. We prove the claim by induction over the length of α. Note that for getting a substring of the form $\sigma\beta\bar{\sigma}$, α must be of the form $\alpha = \rho\gamma\bar{\gamma}$, where $\gamma \neq \varepsilon$ (since strings of the form $\alpha_{l,t}$ cannot contain both characters σ and $\bar{\sigma}$). If $\bar{\gamma}$ is a suffix of $\sigma\beta\bar{\sigma}$, then $\bar{\gamma}$ ends with $\bar{\sigma}$, and does not contain any additional occurences of σ or $\bar{\sigma}$. Therefore, γ starts with σ, and it must be that $\sigma\beta\bar{\sigma} = \gamma\bar{\gamma}$, and in particular β is a palindrome. Else, $\sigma\beta\bar{\sigma}$ is a suffix of $\bar{\gamma}$, therefore $\sigma\bar{\beta}\bar{\sigma}$ is a prefix of γ. In particular, the prefix $\rho\sigma\bar{\beta}\bar{\sigma}$ of $\rho\gamma$ is a proper prefix of α (since $\bar{\gamma} \neq \varepsilon$). Since ρ is a BFB string (Lemma 2 in [36]), the inductive assumption implies that $\bar{\beta}$, and therefore β, is a palindrome. □

Claim 7. *Let α be a BFB string and γ a palindromic concatenation of l-blocks, such that α contains $\bar{\alpha}_{l,t}\gamma\alpha_{l,t}$ as a substring and $top(\gamma) = t' \leq t$. Then, γ is a convexed l-palindrome.*

Proof. By induction on the number of l-blocks composing γ. If γ is composed of zero l-blocks, then $\gamma = \varepsilon$, which is a convexed l-palindrome by definition. Otherwise, γ is of the form $\gamma = \beta_1\beta_2\ldots\beta_q\beta_{q+1}\beta_q\ldots\beta_2\beta_1$, where β_i is an l-block for every $1 \leq i \leq q$, and β_{q+1} is an l-block in case γ is composed of an odd number $2q+1$ of blocks and $\beta_{q+1} = \varepsilon$ in case γ is composed of an even number $2q$ of blocks. Let i be the minimum index such that $top(\beta_i) = t'$. Observe that $\gamma = \gamma'\gamma''\bar{\gamma'}$, where $\gamma' = \beta_1\beta_2\ldots\beta_{i-1}$ is a concatenation of l-blocks such that $top(\gamma') < t'$ (from the selection of i), and $\gamma'' = \beta_i\ldots\beta_q\beta_{q+1}\beta_q\ldots\beta_i$ is a palindromic concatenation of l-blocks with $top(\gamma'') = t'$. Since

γ'' is a substring of α, it is the suffix of some prefix α' of α. From Lemma 2 in [36], α' is a BFB string. From the fact that γ'' starts with $\alpha_{l,t'}$ (as $\alpha_{l,t'}$ is a prefix of the l-t'-block β_i), we get from Claim 5 that γ'' is an l-BFB string, and in particular it is an l-BFB palindrome. In addition, observe that α contains $\bar{\alpha}_{l,t'}\gamma'\alpha_{l,t'} = \bar{\sigma}_{t'}\bar{\alpha}_{l,t'-1}\gamma'\alpha_{l,t'-1}\sigma_{t'}$ as a substring. Since $\bar{\alpha}_{l,t'-1}\gamma'\alpha_{l,t'-1}$ does not contain occurrences of $\sigma_{t'}$ or $\bar{\sigma}_{t'}$, from Claim 6, $\bar{\alpha}_{l,t'-1}\gamma'\alpha_{l,t'-1}$, and in particular γ', is a palindrome. Thus, from the inductive assumption, γ' is a convexed l-palindrome, and by definition $\gamma = \gamma'\gamma''\bar{\gamma}' = \gamma'\gamma''\gamma'$ is a convexed l-palindrome. □

Claim 8. *Let l, t', t be integers such that $l, t' \leq t$. For every convexed l-t'-palindrome γ, $\bar{\alpha}_{l,t} \xrightarrow{BFB} \bar{\alpha}_{l,t}\gamma\alpha_{l,t}$.*

Proof. We prove the claim by induction on t'. When $t' < l$, $\gamma = \varepsilon$ is the only convexed l-t'-palindrome, and by definition $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\alpha_{l,t}$. Otherwise, $t' \geq l$, and assume by induction the claim holds for every l, t'', t such that $t'' < t' \leq t$. By definition, γ is of the form $\gamma'\beta\gamma'$, where γ' is a convexed l-t''-palindrome such that $t'' < t'$, and β is an l-t'-BFB palindrome. From the inductive assumption, $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\gamma'\alpha_{l,t}$, and therefore $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\gamma'\alpha_{l,t'}$. As β is an l-t'-BFB palindrome, β is of the form $\beta = \alpha\bar{\alpha}$, where α is an l-t'-BFB string. In particular, $\alpha_{l,t'} \xrightarrow{\text{BFB}} \alpha$, and from Observation 1 and the fact that $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\gamma'\alpha_{l,t'}$, we get that $\bar{\alpha}_{l,t} \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\gamma'\alpha \xrightarrow{\text{BFB}} \bar{\alpha}_{l,t}\gamma'\alpha\bar{\alpha}\bar{\gamma}'\alpha_{l,t} = \bar{\alpha}_{l,t}\gamma'\beta\gamma'\alpha_{l,t} = \bar{\alpha}_{l,t}\gamma\alpha_{l,t}$. □

Finally, we turn to prove the correctness of Claim 1 from Chapter 4.

Claim 1. *A string α is an l-BFB palindrome if and only if $\alpha = \varepsilon$, α is an l-block, or $\alpha = \beta\gamma\beta$, such that β is an l-BFB palindrome, γ is a convexed l-palindrome, and $\gamma \leq^t \beta$.*

Proof. By definition, if $\alpha = \varepsilon$ or α is an l-block, then α is an l-BFB palindrome. Thus, it remains to show that when α is neither ε nor an l-block, α is an l-BFB palindrome if and only if $\alpha = \beta\gamma\beta$, such that β is an l-BFB palindrome, γ is a convexed l-palindrome, and $\gamma \leq^t \beta$. Let $t = top(\alpha)$.

Assume that α is an l-BFB palindrome which is neither ε nor an l-block. Therefore, α is a concatenation of at least two l-blocks, and so α is of the form $\alpha = \beta\gamma\beta$, such that β is an l-block and γ is some palindromic concatenation of l-blocks. Thus, β must start with the prefix $\alpha_{l,t}$ and end with the suffix $\bar{\alpha}_{l,t}$, and $top(\gamma) \leq t = top(\beta)$. In addition, observe that $\bar{\alpha}_{l,t}\gamma\alpha_{l,t}$ is a substring of α, and from Claim 7, γ is a convexed l-palindrome, proving this direction of the claim.

For the other direction, assume that $\alpha = \beta\gamma\beta$, such that β is an l-BFB palindrome, γ is a convexed l-palindrome, and $\gamma \leq^t \beta$. Therefore, $top(\beta) = t$, and $top(\gamma) = t' \leq t$. Since β is an l-t-BFB string, it starts with the prefix $\alpha_{l,t}$, and being a palindrome it ends with the suffix $\bar{\alpha}_{l,t}$. From Claim 8 and Observation 1, $\beta\gamma\alpha_{l,t}$ is an l-BFB string, and applying again Observation 1, $\beta\gamma\beta = \alpha$ is an l-BFB string. Being a palindrome, α is an l-BFB palindrome. $\square$

C.2 Algorithm SEARCH-BFB

This section completes the missing details in the description of Algorithm SEARCH-BFB in Chapter 4. We describe the FOLD procedure, prove the correctness of the algorithm, and analyze its running time.

C.2.1 Additional Notation and Collection Arithmetics

In order to give an implementation of the FOLD procedure, we first add notation and definitions of some new entities, and observe related properties. For short, from now on we simply say a "collection" when referring to an l-BFB palindrome collection (in some cases we will explicitly indicate that the collection is an l-block collection). A collection containing a single element β will be simply denoted by β, instead of $\{\beta\}$.

For two numbers t, t' and a collection B, $B^{[t,t')}$ denotes the sub-collection containing all elements β in B such that $t \leq top(\beta) < t'$. Denote $B^{\geq t} = B^{[t,\infty)}$ and $B^{<t} = B - B^{\geq t} = B^{[0,t)}$. For a nonempty collection B, denote $min^t(B) = \min_{\beta \in B}\{top(\beta)\}$, where $min^t(\emptyset)$ is defined to be ∞. Say that an element $\beta \in B$ is *minimal* in B if $top(\beta) = min^t(B)$. The collection $B = B' \cap B''$ contains all elements appearing in both B' and B'', where the count of each element $\beta \in B$ equals to the minimum among the counts of β in B' and B''. Say that $B' \subseteq B$ if $B' = B \cap B'$. Notations of the form $\vec{a}$ will denote series $\vec{a} = a_0, a_1, a_2, \ldots$, and $\vec{a}_d$ denotes the prefix $a_0, a_1, \ldots, a_d$ of $\vec{a}$. For an integer

$m \neq 0$, denote by d_m the maximum integer $d \geq 0$ such that m is divided by 2^d. For example, $d_8 = d_{-24} = 3$, and $d_7 = 0$. Observe that $d_m = 0$ when m is odd, and otherwise $d_m = 1 + d_{\frac{m}{2}}$. d_m can also be understood as the index of the least significant bit different from 0 in the binary representation of m, and in particular $d_m \leq \log_2 m$.

Observation 2. *For two collections B, B',*

- $$\begin{aligned} \text{mod2}(B+B') = {} & \text{mod2}(B + \text{mod2}(B')) = \text{mod2}(\text{mod2}(B) + B') \\ = {} & \text{mod2}(\text{mod2}(B) + \text{mod2}(B')) \\ = {} & \text{mod2}(B) + \text{mod2}(B') - 2(\text{mod2}(B) \cap \text{mod2}(B')) \end{aligned}.$$

- *For an integer $i \geq 0$,* $\text{mod2}(B + iB') = \text{mod2}(B + B')$ *when i is odd, and* $\text{mod2}(B + iB') = \text{mod2}(B)$ *when i is even. In particular,* $\text{mod2}(B - B') = \text{mod2}(B - B' + 2B') = \text{mod2}(B + B')$.

- *For two integers t and t',* $\text{mod2}\left(B^{[t,t')}\right) = (\text{mod2}(B))^{[t,t')}$.

Definition 4. *A* convexed *l*-collection of order *q is an l-BFB palindrome collection A of the form $A = \{\alpha_1, 2\alpha_1, 4\alpha_2, \ldots, 2^{q-1}\alpha_q\}$, where $\alpha_q <^t \alpha_{q-1} <^t \ldots <^t \alpha_1$.*

A convexed l-collection of order q $A = \{\alpha_1, \ldots, 2^{q-1}\alpha_q\}$ satisfies $|A| = 2^q - 1$. In addition, $A = \emptyset$ when $q = 0$, and when $A \neq \emptyset$, $\text{mod2}(A) = \alpha_1$ and $\frac{A}{2} \equiv \frac{1}{2}A = \{\alpha_2, 2\alpha_3, \ldots, 2^{q-2}\alpha_q\}$ is a convexed l-collection of order $q-1$. It is possible to concatenate all elements in A to produce a convexed l-palindrome γ_A, where $\gamma_A = \varepsilon$ if $A = \emptyset$, and otherwise $\gamma_A = \gamma_{\frac{A}{2}} \alpha_1 \gamma_{\frac{A}{2}}$. In Fig. 2a, all 1-blocks besides the two repeats of β_1 form a convexed 1-collection $A = \{\beta_2, 2\beta_3, 4\beta_4\}$ of order 3, where $\gamma_A = \beta_4\beta_3\beta_4\beta_2\beta_4\beta_3\beta_4$.

Observation 3. *For a convexed l-collection A and an integer m,* $|\text{mod2}(mA)| = 0$ *if either m is even or $A = \emptyset$, and otherwise* $|\text{mod2}(mA)| = 1$.

Claim 9. *Let $A = \{\alpha_1, \ldots, 2^{j-1}\alpha_j, \ldots, 2^{r-1}\alpha_r\}$ and $A' = \{\alpha_1, \ldots, 2^{j-1}\alpha_j\}$ be two convexed l-collections (where $A' \subseteq A$, and it is possible that $A' = \emptyset$). For every number t, there is an integer $x \geq 0$ and a convexed l-collection $\hat{A}$ such that $(A - A')^{<t} = 2^x\hat{A}$. In addition, if $A' \neq \emptyset$ then $x > 0$ and $|\hat{A}| < |A|$*

Proof. First, note that $A - A' = \{2^j\alpha_{j+1}, \ldots, 2^{r-1}\alpha_r\}$. Now, let $x = j$ if $top\left(\alpha_{j+1}\right) < t$, and otherwise let x be the maximum integer in the range $j < x \leq r$ such that $top(\alpha_x) \geq$

Table C.1: The decomposition and signature of the collection $B = \{2\beta_1, \beta_2, 2\beta_3, 4\beta_4\}$. Here, $r(B) = 3$.

d	B_d	L_d	H_d	s_d
0	$\{2\beta_1, \beta_2, 2\beta_3, 4\beta_4\}$	β_2	$2\beta_1$	1
1	$\{\beta_3, 2\beta_4\}$	β_3	$\emptyset$	0
2	β_4	β_4	$\emptyset$	0
3	$\emptyset$	$\emptyset$	$\emptyset$	-1
4	$\emptyset$	$\emptyset$	$\emptyset$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

t. Then, $(A - A')^{<t} = \{2^x \alpha_{x+1}, \dots, 2^{r-1}\alpha_r\} = 2^x\{\alpha_x, \dots, 2^{r-x-1}\alpha_r\}$. Choosing $\hat{A} = \{\alpha_x, \dots, 2^{r-x-1}\alpha_r\}$, the claim follows. □

Definition 5. *Let $B = \{n_1\beta_1, n_2\beta_2, \dots, n_q\beta_q\}$ be an l-BFB palindrome collection. The* decomposition *of B is a series triplet $\left\langle \vec{B}, \vec{L}, \vec{H} \right\rangle$, whose elements are recursively defined as follows:*

- *$B_0 = B$, and $B_d = \frac{1}{2}(B_{d-1} - L_{d-1} - H_{d-1})$ for $d > 0$.*
- *$L_d = \text{mod2}(B_d)$.*
- *$H_d = (B_d - L_d)^{\geq \min^t(L_d)}$.*

Denote by $r(B)$ the minimum integer r such that $B_r = \emptyset$.

Table C.1 gives the decomposition of the collection B^1 corresponding to Fig. 2a in Chapter 4. In what follows, let B be a collection, $\left\langle \vec{B}, \vec{L}, \vec{H} \right\rangle$ its decomposition, and $r = r(B)$.

By definition, $L_d, H_d \subseteq B_d$. It may be observed that the count of each element in L_d is exactly 1 (by definition of the $\text{mod2}(\cdot)$ operation), i.e. $\text{mod2}(L_d) = L_d$, the count of each element in $B_d - L_d$ is even (since reducing L_d from B_d decreases by 1 the count of each element with an odd count in B_d), therefore the counts of all elements in H_d and in $B_d - L_d - H_d$ are even (since nonzero counts in these collections equal to the corresponding even counts in $B_d - L_d$), i.e. $\text{mod2}(B_d - L_d) = \text{mod2}(H_d) = \text{mod2}(B_d - L_d - H_d) = \emptyset$. In addition, every single occurrence of an element $\beta \in B_d$ (and in particular every $\beta \in H_d$ or $\beta \in L_d$), corresponds to 2^d repeats of β in B.

Definition 6. *For a collection B, $\vec{t}(B) = \vec{t}$ is the non-decreasing series of numbers whose elements are given by $t_0 = \infty$, and $t_d = \min(\min^t(L_d), t_{d-1}))$ for $d > 0$.*

The following observation may be easily asserted, in an inductive manner.

Observation 4. *For a collection B and every integer $d \geq 0$, $H_d = (B_d - L_d)^{\geq t_{d+1}}$, $B^{<t_d} = 2^d B_d$, and $B^{[t_{d+1}, t_d)} = 2^d(L_d + H_d)$.*

Finally, we define the *signature* of a collection, which is derived from its decomposition and will serve as an optimality measure implying the folding restrictions over the collection.

Definition 7. *The* signature *of B is a series $\vec{s} = \vec{s}(B)$, where $s_0 = |L_0|$, and $s_d = |L_d| - |L_{d-1}| - \frac{|H_{d-1}|}{2} + \max(s_{d-1}, 0)$ for $d > 0$.*

The last column of Table C.1 shows the signature of the exemplified collection. For two signatures $\vec{s} = s_0, s_1, \ldots$ and $\vec{s}' = s'_0, s'_1, \ldots$, denote $\vec{s} < \vec{s}'$ if $\vec{s}$ precedes $\vec{s}'$ lexicographically, i.e. there is some integer $d \geq 0$ such that $s_i = s'_i$ for every $0 \leq i < d$, and $s_d < s'_d$. Denote $\vec{s} \leq \vec{s}'$ if $\vec{s} < \vec{s}'$ or $\vec{s} = \vec{s}'$. We will show that signatures can serve as an optimality measure for collections, where lower signature collections are always less restricted than higher signature collection with respect to folding possibilities.

From now on, when discussing derived entities such a decompositions $\left\langle \vec{B}, \vec{L}, \vec{H} \right\rangle$, signatures $\vec{s}$, etc., we assume these entities correspond to the collection B discussed in the same context without stating so explicitly. When several collections are considered, these collections are annotated with superscripts (e.g. B', B^*, B^3, etc.), which also annotate their correspondingly derived entities (e.g. $L'_d, \vec{s}^3$, etc.).

Claim 10. *For every $d \geq 0$, $|L_d| \geq \max(s_d, 0)$ and $|B_d| + |L_d| - s_d \geq \max(s_d, 0)$.*

Proof. We first show the first inequality in the Claim. For $d = 0$, $s_0 = |L_0| \geq 0$ by definition. Assume by induction $|L_{d'}| \geq \max(s_{d'}, 0)$ for every $0 \leq d' < d$. Then, $|L_d| = s_d + |L_{d-1}| + \frac{|H_{d-1}|}{2} - \max(s_{d-1}, 0) \geq s_d + \frac{|H_{d-1}|}{2} \geq s_d$. In addition, $|L_d| \geq 0$, and so $|L_d| \geq \max(s_d, 0)$. The second inequality follows immediately from the first one, as $|B_d| + |L_d| - s_d \geq |B_d| \geq |L_d| \geq \max(s_d, 0)$. □

Claim 11. *For $r = r(B)$, $s_r = -\frac{|B_{r-1}| + |L_{r-1}|}{2} + \max(s_{r-1}, 0) \leq 0$, and $s_d = 0$ for every $d > r$.*

Proof. The inequality $s_r \leq 0$ follows immediately from Claim 10 and the fact that $|L_r| = 0$. In addition, since $B_r = \frac{1}{2}(B_{r-1} - L_{r-1} - H_{r-1}) = \emptyset$, we have that $H_{r-1} = B_{r-1} - L_{r-1}$, and therefore $s_r = |L_r| - |L_{r-1}| - \frac{|H_{r-1}|}{2} + \max(s_{r-1}, 0) = -\frac{|B_{r-1}|+|L_{r-1}|}{2} + \max(s_{r-1}, 0)$.

To show the second part of the claim, note that $|B_d| = |L_d| = |H_d| = 0$ for every $d \geq r$. This implies that for every $d > r$, $s_d = |L_d| - |L_{d-1}| - \frac{|H_{d-1}|}{2} + \max(s_{d-1}, 0) = \max(s_{d-1}, 0)$. Since we showed that $s_r \leq 0$, we have that $s_{r+1} = \max(s_r, 0) = 0$, and inductively it follows that that $s_d = 0$ for every $d > r$. □

Define the series $\vec{\Delta} = \vec{\Delta}(B)$, where $\Delta_0 = 0$, and $\Delta_d = \Delta_{d-1} + 2^{d-1}\text{abs}(s_{d-1})$ for $d > 0$ (where $\text{abs}(s_{d-1})$ is the absolute value of s_{d-1}).

Claim 12. *For every integer $d \geq 0$,*

$$|B| = 2^d \left(|B_d| + |L_d| - s_d\right) + \Delta_d \geq 2^d \max(s_d, 0) + \Delta_d.$$

Proof. The fact that $2^d \left(|B_d| + |L_d| - s_d\right) + \Delta_d \geq 2^d \max(s_d, 0) + \Delta_d$ follows from Claim 10. The equality $|B| = 2^d \left(|B_d| + |L_d| - s_d\right) + \Delta_d$ is proven by induction on d. For $d = 0$, since $B_0 = B$, $s_0 = |L_0|$, and $\Delta_0 = 0$ by definition, we get that $2^0 \left(|B_0| + |L_0| - s_0\right) + \Delta_d = |B|$. Now, assuming the claim holds for some $d \geq 0$, we show it also holds for $d' = d + 1$:

$$\begin{aligned}
|B| &= 2^d \left(|B_d| + |L_d| - s_d\right) + \Delta_d \\
&= 2^d \left(|B_d| + |L_d| - s_d - \text{abs}(s_d)\right) + \Delta_{d+1} \\
&= 2^d \left(|B_d| + |L_d| - 2\max(s_d, 0)\right) + \Delta_{d+1} \\
&= 2^d \left((2|B_{d+1}| + |L_d| + |H_d|) + |L_d| - 2\max(s_d, 0)\right) + \Delta_{d+1} \\
&= 2^d \left(2|B_{d+1}| + 2|L_{d+1}| - 2s_{d+1}\right) + \Delta_{d+1} \\
&= 2^{d+1} \left(|B_{d+1}| + |L_{d+1}| - s_{d+1}\right) + \Delta_{d+1}.
\end{aligned}$$

□

Conclusion 1. *For every $d \geq r = r(B)$ we have that $|B_d| = |L_d| = 0$, where from Claim 11 $s_d = 0$ for $d > r$. Thus, Claim 12 implies that $|B| = -2^r s_r + \Delta_r = \Delta_d$ for every $d > r$.*

Claim 13. *Let B and B' be two collections such that $|B| = |B'| = n$ and $\vec{s}_{r-1} = \vec{s}'_{r-1}$ for $r = r(B)$. Then, $\vec{s} \leq \vec{s}'$.*

Proof. Since $\vec{s}_{r-1} = \vec{s}'_{r-1}$, it follows that $\Delta_r = \Delta'_r$. From Conclusion 1, $s_r = \frac{\Delta_r - n}{2^r}$. From Claim 12, $s'_r = \frac{\Delta'_r - n}{2^r} + |B'_r| + |L'_r| \geq \frac{\Delta_r - n}{2^r} = s_r$. If $s'_r > s_r$, then $\vec{s} < \vec{s}'$ and the claim holds. If $s'_r = s_r$, then in particular $|B'_r| = 0$ and so $B'_r = \emptyset$. Thus, $r(B') \leq r$, and so for every $d > r$ we have that $s'_d = s_d = 0$, and $\vec{s}' = \vec{s}$. □

C.2.2 Folding Increases Signature

This section is dedicated for proving the following claim:

Claim 14. *Let B' be a folding of an l-block collection B. If $B \neq B'$, then $\vec{s} < \vec{s}'$.*

The proof, given at the end of this section, is based on an observation that shows how to present a general folding as a series of a special kind of elementary foldings, and showing that such elementary foldings always increase the signature of the collection.

Definition 8. *Let B' be a folding of B.*

- *Say that B' is a* type I elementary folding *if B' is of the form $B' = B - m(2\beta + A) + m\alpha$, where β is an l-block, $A \neq \emptyset$ is a convexed l-collection, $m > 0$ is an integer such that $m(2\beta + A) \subseteq B$, and $\alpha = \beta\gamma_A\beta$ is an l-BFB palindrome such that $\alpha \notin B$.*
- *Say that B' is a* type II elementary folding *if $\varepsilon \notin B$ and B' is of the form $B' = B + m\varepsilon$.*

Claim 15. *Let B be a collection of l-blocks. For every folding B' of B there is a sequence of collections $B^0, B^1, \ldots, B^j$, where $B^0 = B'$, $B^j = B$, and for every $0 \leq i < j$, B^i is a (type I or II) elementary folding of B^{i+1}.*

Proof. By definition, each element in a folding B' of B is either an l-block from B, a concatenation of several l-blocks from B, or ε. The sequence $B^0, B^1, \ldots, B^j$ is built iteratively as follows.

Initiate $B^0 = B'$, and $i = 0$. As long as $B^i \neq B$, we show how to compute B^{i+1} given the collection B^i. The construction maintains the property that each computed collection B^i is a folding of B. In the case where B^i contains some composite l-BFB palindrome of the form $\alpha = \beta\gamma_A\beta$, let m be the count of α in B^i, and set $B^{i+1} = B^i - m\alpha + m(2\beta + A)$. We may assume $A \neq \emptyset$, since when $\gamma_A = \varepsilon$ we can choose $A = \varepsilon$.

Observe that B^{i+1} is a folding of B (where the same sub-collection of l-blocks from B which composes the m copies of α in B^i, composes the elements in the m repeats of $2\beta + A$ in B^{i+1}), and that B^i is a type I elementary folding of B^{i+1}. In addition, since the number of l-blocks composing each element in A is less than the number of l-blocks composing α, after a finite number of such modification there will be no more composite palindromes in the collection.

In the case where B^i contains no composite palindrome, B^i is a folding of B containing only l-blocks and 0 or more ε elements. If B^i contains no ε elements, then $B^i = B$, and the process is completed choosing $j = i$. Else, for m the count of ε in B^i, set $B^{i+1} = B^i - m\varepsilon$, and therefore B^i is a type II elementary folding of B_{i+1}. Note that $B^{i+1} = B$, completing the process for $j = i+1$. □

Observe that the signature of a collection depends only in its decomposition, and is independent in the manner the collection was obtained. Therefore, from the above claim, in order to show that foldings necessarily increase signatures, it is enough to show that each elementary folding increases the signature. In what follows, we give several technical claims that will prove this property.

Claim 16. *Let B, B', and A be l-BFB palindrome collections and $m > 0$ an integer such that $B' = B + mA$. Then,*

1. *For every $0 \leq i \leq d_m$, $B'_i = B_i + \frac{m}{2^i}A^{<t_i}$.*
2. *For every $0 \leq i < d_m$, $L'_i = L_i$ (i.e. $\vec{L}'_{d_m-1} = \vec{L}_{d_m-1}$).*
3. *For every $0 \leq i < d_m$, $H'_i = H_i + \frac{m}{2^i}A^{[t_{i+1},t_i)}$.*

Proof. $B_0 = B$ and $t_0 = \infty$, therefore $B'_0 = B' = B + mA = B_0 + \frac{m}{2^0}A^{<t_0}$. Thus, the first item in the claim holds for $i = 0$, and the two other items hold trivially for every $0 \leq i < 0$.

Assuming by induction that for some $i < d_m$ the first item holds for every $0 \leq i' \leq i$ and the two other items hold for every $0 \leq i' < i$, we show that (1) $B'_{i+1} = B_{i+1} + \frac{m}{2^{i+1}}A^{<t_{i+1}}$, (2) $L'_i = L_i$, and (3) $H'_i = H_i + \frac{m}{2^i}A^{[t_{i+1},t_i)}$.

We start by showing (2). Since $i < d_m$, $\frac{m}{2^i}$ is even, and so $L'_i = \text{mod2}\left(B'_i\right) = \text{mod2}\left(B_i + \frac{m}{2^i}A^{<t_i}\right) \overset{\text{Obs.2}}{=} \text{mod2}\left(B_i\right) = L_i$. In order to show (3), note that $\vec{L}'_i = \vec{L}_i$ implies that $t'_{i+1} = t_{i+1}$, therefore $H'_i \overset{\text{Obs.4}}{=} (B'_i - L'_i)^{\geq t'_{i+1}} = (B_i + \frac{m}{2^i}A^{<t_i} - L_i)^{\geq t_{i+1}} =$

$(B_i - L_i)^{\geq t_{i+1}} + \frac{m}{2^i}A^{[t_{i+1},t_i)} \overset{\text{Obs.4}}{=} H_i + \frac{m}{2^i}A^{[t_{i+1},t_i)}$. Finally, (1) is true since $B'_{i+1} = \frac{1}{2}(B'_i - L'_i - H'_i) = \frac{1}{2}(B_i + \frac{m}{2^i}A^{<t_i} - L_i - H_i - \frac{m}{2^i}A^{[t_{i+1},t_i)}) = \frac{1}{2}(B_i - L_i - H_i) + \frac{m}{2^{i+1}}(A^{<t_i} - A^{[t_{i+1},t_i)}) = B_{i+1} + \frac{m}{2^{i+1}}A^{<t_{i+1}}$. □

Claim 17. *Let $B' = B - m(2\beta + A) + m\alpha$ be a type I elementary folding of B. Then,*

$$\forall 0 \leq i \leq d_m,\ B'_i = B_i - \frac{m}{2^i}(2\beta + A)^{<t_i} + \frac{m}{2^i}\alpha^{<t_i}, \tag{C.1}$$

$$\vec{L}'_{d_m-1} = \vec{L}_{d_m-1} \tag{C.2}$$

$$\forall 0 \leq i < d_m,\ H'_i = H_i - \frac{m}{2^i}(2\beta + A)^{[t_{i+1},t_i)} + \frac{m}{2^i}\alpha^{[t_{i+1},t_i)}. \tag{C.3}$$

Proof. Let $B'' = B - m(2\beta + A)$. Therefore, $B = B'' + m(2\beta + A)$, and $B' = B'' + m\alpha$. From Claim 16,

1. For every $0 \leq i \leq d_m$, $B_i = B''_i + \frac{m}{2^i}(2\beta + A)^{<t_i}$, and $B'_i = B''_i + \frac{m}{2^i}\alpha^{<t_i}$. Therefore, $B'_i = B_i - \frac{m}{2^i}(2\beta + A)^{<t_i} + \frac{m}{2^i}\alpha^{<t_i}$.
2. $\vec{L}'_{d_m-1} = \vec{L}_{d_m-1} = \vec{L}''_{d_m-1}$.
3. For every $0 \leq i < d_m$, $H_i = H''_i + \frac{m}{2^i}(2\beta + A)^{[t_{i+1},t_i)}$, and $H'_i = H''_i + \frac{m}{2^i}\alpha^{[t_{i+1},t_i)}$. Therefore, $H'_i = H_i - \frac{m}{2^i}(2\beta + A)^{[t_{i+1},t_i)} + \frac{m}{2^i}\alpha^{[t_{i+1},t_i)}$.

□

Claim 18. *Let B, B', and C be l-BFB palindrome collections, A a convexed l-collection, and m and i two nonnegative integers, such that* (a) $\vec{s}'_i = \vec{s}_i$, (b) $t'_{i+1} \geq t_{i+1}$, (c) $|L'_i| + \frac{|H'_i|}{2} < |L_i| + \frac{|H_i|}{2} - |C|$, (d) $B_{i+1} \cap C = \emptyset$, *and* (e) $B'_{i+1} = B_{i+1} + C - mA$. *Then, $B <^s B'$.*

Proof. We prove the claim by induction on the size of A. Let $A = \{\alpha_1, \ldots, 2^{r-1}\alpha_r\}$, and denote $\tilde{A} = \text{mod2}(mA)$. Observe that $\tilde{A} = \alpha_1$ when $A \neq \emptyset$ and m is odd, and otherwise $\tilde{A} = \emptyset$. In addition, observe that $\text{mod2}(B_{i+1} + C) \overset{\text{Obs.2}}{=} \text{mod2}(B_{i+1}) + \text{mod2}(C) - 2(\text{mod2}(B_{i+1}) \cap \text{mod2}(C)) \overset{\text{(d)}}{=} \text{mod2}(B_{i+1}) + \text{mod2}(C) = L_{i+1} + \text{mod2}(C)$. Therefore,

$$\begin{aligned}
L'_{i+1} = {} & \operatorname{mod2}\left(B'_{i+1}\right) \overset{(e)}{=} \operatorname{mod2}\left(B_{i+1}+C-mA\right) \\
\overset{\text{Obs.2}}{=} {} & \operatorname{mod2}\left(B_{i+1}+C\right)+\operatorname{mod2}\left(mA\right)-2\left(\operatorname{mod2}\left(B_{i+1}+C\right)\cap\operatorname{mod2}\left(mA\right)\right) \\
= {} & L_{i+1}+\operatorname{mod2}\left(C\right)+\tilde{A}-2\left(\left(L_{i+1}+\operatorname{mod2}\left(C\right)\right)\cap\tilde{A}\right)
\end{aligned}$$

Since $(L_{i+1}+\operatorname{mod2}(C))\cap\tilde{A}\subseteq\tilde{A}$, it follows that $|L'_{i+1}|\geq|L_{i+1}|+|\operatorname{mod2}(C)|+|\tilde{A}|-2|\tilde{A}|=|L_{i+1}|+|\operatorname{mod2}(C)|-|\tilde{A}|$. Therefore, $s'_{i+1}=|L'_{i+1}|-|L'_i|-\frac{|H'_i|}{2}+\max(s'_i,0)\overset{(a),(c)}{>}|L_{i+1}|+|\operatorname{mod2}(C)|-|\tilde{A}|-|L_i|-\frac{|H_i|}{2}+|C|+\max(s_i,0)=s_{i+1}+|\operatorname{mod2}(C)|+|C|-|\tilde{A}|$. As both s'_{i+1} and s_{i+1} are integers, $s'_{i+1}\geq s_{i+1}+|\operatorname{mod2}(C)|+|C|-|\tilde{A}|+1\geq s_{i+1}$ (recall that $|\tilde{A}|\leq 1$). When $s'_{i+1}>s_{i+1}$, $\vec{s}<\vec{s}'$ and the claim follows. Otherwise, $s'_{i+1}=s_{i+1}$, and there is a need to continue and examine positions grater than $i+1$ in the signatures of B and B'.

Note that for obtaining $s'_{i+1}=s_{i+1}$ we must have that $C=\emptyset$ and $\tilde{A}=L_{i+1}\cap\tilde{A}=\alpha_1$, which implies that $A\neq\emptyset$, m is odd, and $L'_{i+1}=L_{i+1}+\alpha_1-2\alpha_1=L_{i+1}-\alpha_1$ (thus $\alpha_1\in L_{i+1}$, and in particular $top(\alpha_1)\geq min^t(L_{i+1})\geq t_{i+2}$). Assuming by induction that the claim holds for every B'',B''',C',A',i', and m' sustaining requirements (a) to (e) and $|A'|<|A|$, we show the claim also holds for B,B',C,A,i, and m.

Now, since $\vec{s}'_i\overset{(a)}{=}\vec{s}_i$ and $s'_{i+1}=s_{i+1}$, requirement (a) in the claim holds with respect to $i'=i+1$. In addition,

$$\begin{aligned}
t'_{i+2} = {} & \min(min^t(L'_{i+1}),t'_{i+1})\geq\min(min^t(L_{i+1}-\alpha_1),t_{i+1}) \\
\geq {} & \min(min^t(L_{i+1}),t_{i+1})=t_{i+2},
\end{aligned}$$

thus requirement (b) also holds with respect to $i'=i+1$. Furthermore,

$$\begin{aligned}
H'_{i+1} \overset{\text{Obs.4}}{=} {} & (B'_{i+1}-L'_{i+1})^{\geq t'_{i+2}}=(B_{i+1}-mA-L_{i+1}+\alpha_1)^{\geq t'_{i+2}} \\
= {} & (B_{i+1}-L_{i+1})^{\geq t'_{i+2}}-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}} \\
= {} & \left((B_{i+1}-L_{i+1})^{\geq t_{i+2}}-(B_{i+1}-L_{i+1})^{\left[t_{i+2},t'_{i+2}\right)}\right)-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}} \\
= {} & H_{i+1}-(B_{i+1}-L_{i+1})^{\left[t_{i+2},t'_{i+2}\right)}-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}}.
\end{aligned}$$

Since $\alpha_1\in A$, the operation $A-\alpha_1$ yields a valid collection. In addition, note that the count of each element in $B_{i+1}-L_{i+1}$ is even, and since m is odd, B_{i+1} contains at least m copies of α_1 (as $mA\subseteq B_{i+1}$) and L_{i+1} contains exactly one copy of α_1, $B_{i+1}-$

$L_{i+1}-(m-1)\alpha_1$ is a valid collection, in which the count of each element is even. Denote $\hat{C}=\frac{1}{2}\left((B_{i+1}-L_{i+1}-(m-1)\alpha_1)^{[t_{i+2},t'_{i+2})}\right)=$
$\frac{1}{2}\left((B_{i+1}-L_{i+1})^{[t_{i+2},t'_{i+2})}-(m-1)\alpha_1{}^{<t'_{i+2}}\right)$. Now we can write

$$\begin{aligned}
H'_{i+1} &= H_{i+1}-(B_{i+1}-L_{i+1})^{[t_{i+2},t'_{i+2})}-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}}\\
&= H_{i+1}-2\hat{C}-(m-1)\alpha_1{}^{<t'_{i+2}}-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}}\\
&= H_{i+1}-2\hat{C}-(m-1)\left(\alpha_1-\alpha_1{}^{\geq t'_{i+2}}\right)-mA^{\geq t'_{i+2}}+\alpha_1{}^{\geq t'_{i+2}}\\
&= H_{i+1}-2\hat{C}-(m-1)\alpha_1-m(A-\alpha_1)^{\geq t'_{i+2}}
\end{aligned}$$

From the above $|H'_{i+1}|\leq|H_{i+1}|-2|\hat{C}|$, and since $|L'_{i+1}|=|L_{i+1}|-1$ we get that $|L'_{i+1}|+\frac{|H'_{i+1}|}{2}\leq|L'_{i+1}|-1+\frac{|H'_{i+1}|}{2}+|\hat{C}|<|L'_{i+1}|+\frac{|H'_{i+1}|}{2}+|\hat{C}|$, and in particular requirement (c) holds with respect to $C'=\hat{C}$, and $i'=i+1$. Moreover, by definition the top values of all elements in $\hat{C}$ are at least t_{i+2}, and from Observation 4, $B_{i+2}=\frac{1}{2^{i+2}}B^{<t_{i+2}}$, hence the top values of all elements in B_{i+2} are lower than t_{i+2}. Thus, $B_{i+2}\cap\hat{C}=\emptyset$, and requirement (d) holds with respect to $C'=\hat{C}$, and $i'=i+1$.

From Claim 9, there is an integer $x>0$ and a convex l-collection $\hat{A}$ such that $|\hat{A}|<|A|$ and $(A-\alpha_1)^{<t'_{i+2}}=2^x\hat{A}$. Therefore,

$$\begin{aligned}
H'_{i+1} &= H_{i+1}-2\hat{C}-(m-1)\alpha_1-m(A-\alpha_1)^{\geq t'_{i+2}}\\
&= H_{i+1}-2\hat{C}-(m-1)\alpha_1-m\left((A-\alpha_1)-(A-\alpha_1)^{<t'_{i+2}}\right)\\
&= H_{i+1}-2\hat{C}-mA+\alpha_1+m(A-\alpha_1)^{<t'_{i+2}}\\
&= H_{i+1}-2\hat{C}-mA+\alpha_1+m2^x\hat{A},
\end{aligned}$$

and

$$\begin{aligned}
B'_{i+2} &= \tfrac{1}{2}(B'_{i+1}-L'_{i+1}-H'_{i+1})\\
&= \tfrac{1}{2}((B_{i+1}-mA)-(L_{i+1}-\alpha_1)-(H_{i+1}-2\hat{C}-mA+\alpha_1+m2^x\hat{A}))\\
&= \tfrac{1}{2}(B_{i+1}-L_{i+1}-H_{i+1})+\hat{C}-m2^{x-1}\hat{A}\\
&= B_{i+2}+\hat{C}-m2^{x-1}\hat{A}.
\end{aligned}$$

Since $x>0$, $m2^{x-1}$ is an integer. Therefore, requirement (e) holds with respect to $C'=\hat{C},A'=\hat{A},\ i'=i+1$, and $m'=m2^{x-1}$. From the inductive assumption and the fact that $|\hat{A}|<|A|$, the claim follows. □

Claim 19. *Let B' be a type I elementary folding of B. Then, $B <^s B'$*

Proof. By definition, B' is of the form $B' = B - m(2\beta + A) + m\alpha$, where $m > 0$ is an integer, α and β are l-BFB palindromes, and $A = \{\alpha_1, \ldots, 2^{r-1}\alpha_r\}$ is a convexed l-collection such that $\alpha = \beta\gamma_A\beta$. Let $q \geq 0$ be the index such that $\beta \in L_q + H_q$. From Observation 4, $t_{q+1} \leq top(\beta) < t_q$, and therefore for every $0 \leq i \leq q$ and every $\alpha_j \in A$ we have that $(\star)$ $top\left(\alpha_j\right) \leq top(\alpha) = top(\beta) < t_q \leq t_i$. Let $d = d_m$. We consider two cases: (1) $q < d$, and (2) $q \geq d$, and show for each case that $B <^s B'$.

(1) $q < d$. In this case, condition $(\star)$ implies that for every $0 \leq i < q$, we have that $(2\beta + A)^{[t_{i+1}, t_i)} = \alpha^{[t_{i+1}, t_i)} = \emptyset$. In addition $(2\beta + A)^{\left[t_{q+1}, t_q\right)} = 2\beta + A^{\geq t_{q+1}}$, $\alpha^{\left[t_{q+1}, t_q\right)} = \alpha$, and $(2\beta + A)^{<t_{q+1}} = A^{<t_{q+1}}$, $\alpha^{<t_{q+1}} = \emptyset$. Thus, form Claim 17, we get that

$$\begin{aligned}
B'_{q+1} &= B_{q+1} - \tfrac{m}{2^{q+1}} A^{<t_{q+1}} \overset{\text{Clm.9}}{=} B_{q+1} - \tfrac{m2^x}{2^{q+1}} \hat{A}, \\
&\quad \text{where } \tfrac{m2^x}{2^{q+1}} \text{ is a positive integer and } \hat{A} \text{ is a convexed } l\text{-collection,} \\
\vec{L}'_q &= \vec{L}_q, \\
\vec{H}'_{q-1} &= \vec{H}_{q-1}, \\
H'_q &= H_q - \tfrac{m}{2^q}(2\beta + A^{\geq t_{q+1}}) + \tfrac{m}{2^q}\alpha.
\end{aligned}$$

Observe that $\vec{L}'_q = \vec{L}_q$ and $\vec{H}'_{q-1} = \vec{H}_{q-1}$ imply that $\vec{s}'_q = \vec{s}_q$ and $\vec{t}'_{q+1} = \vec{t}_{q+1}$. Also, observe that $|H'_q| \leq |H_q| - \frac{m}{2^q} < |H_q|$, therefore $|L'_q| + \frac{|H'_q|}{2} < |L_q| + \frac{|H_q|}{2}$. Applying Claim 18 with respect to entities $B, B', C = \emptyset, \hat{A}, i = q$, and $m' = \frac{m2^x}{2^{q+1}}$, we get that $B <^s B'$.

(2) $q \geq d$. In this case, condition $(\star)$ implies that for every $0 \leq i < d$, we have that $(2\beta + A)^{[t_{i+1}, t_i)} = \alpha^{[t_{i+1}, t_i)} = \emptyset$, and $(2\beta + A)^{<t_d} = 2\beta + A$, $\alpha^{<t_d} = \alpha$. Therefore, from Claim 17,

$$\begin{aligned}
B'_d &= B_d - \tfrac{m}{2^d}(2\beta + A) + \tfrac{m}{2^d}\alpha, \\
\vec{L}'_{d-1} &= \vec{L}_{d-1}, \\
\vec{H}'_{d-1} &= \vec{H}_{d-1}.
\end{aligned}$$

Again, $\vec{L}'_{d-1} = \vec{L}_{d-1}$ and $\vec{H}'_{d-1} = \vec{H}_{d-1}$ imply that $\vec{t}'_d = \vec{t}_d$ and $\vec{s}'_{d-1} = \vec{s}_{d-1}$. Denote $m' = \frac{m}{2^d}$, and observe that m' is an odd nonnegative integer. Thus, $\text{mod2}\left(m'(2\beta + A)\right) \overset{\text{Obs.2}}{=} \text{mod2}(A) = \alpha_1$, and $\text{mod2}(m'\alpha) \overset{\text{Obs.2}}{=} \alpha$. Since $\alpha \notin B_d -$

$m'(2\beta + A) \subseteq B$ (by definition of elementary folding), we get that $\mathrm{mod2}\left(B_d - m'(2\beta + A)\right) \cap \mathrm{mod2}\left(m'\alpha\right) = \emptyset$. Consequentially,

$$\begin{aligned} L'_d = & \ \mathrm{mod2}\left(B'_d\right) = \mathrm{mod2}\left(B_d - m'(2\beta + A) + m'\alpha\right) \\ \overset{\text{Obs.2}}{=} & \ L_d + \alpha_1 - 2(L_d \cap \alpha_1) + \alpha. \end{aligned}$$

Note that $|L'_d| = |L_d| + 2 - 2|L_d \cap \alpha_1|$, and therefore $s'_d = |L'_d| - |L'_{d-1}| - \frac{|H'_{d-1}|}{2} + \max(s'_{d-1}, 0) = |L_d| + 2 - 2|L_d \cap \alpha_1| - |L_{d-1}| - \frac{|H_{d-1}|}{2} + \max(s_{d-1}, 0) = s_d + 2 - 2|L_d \cap \alpha_1|$. When $L_d \cap \alpha_1 = \emptyset$, $s'_d = s_d + 2$, and so $\vec{s} < \vec{s}'$ and the claim follows. Else, $L_d \cap \alpha_1 = \alpha_1$, $L'_d = L'_d - \alpha_1 + \alpha$, therefore $s'_d = s_d$, and there is a need to continue and examine positions grater than d in the signatures of B' and B.

In this remaining case $\vec{s}'_d = \vec{s}_d$, thus requirement (a) in Claim 18 holds with respect to B, B' and $i = d$. In addition, $\alpha_1 \in L_d$, implying that $t_{d+1} \leq min^t(L_d) \leq top(\alpha_1) \leq top(\beta)$, and so $q = d$. Moreover, $t'_{d+1} \leq min^t(L'_d) \leq top(\alpha) = top(\beta)$, and $t'_{d+1} = \min(min^t(L'_d), t'_d) = \min(min^t(L_d - \alpha_1 + \alpha), t_d) \geq \min(min^t(L_d), t_d) = t_{d+1}$, hence requirement (b) in Claim 18 holds with respect to B, B' and $i = d$. Now,

$$\begin{aligned} H'_d = & \ (B'_d - L'_d)^{\geq t'_{d+1}} \\ = & \ \left((B_d - m'(2\beta + A) + m'\alpha) - (L_d - \alpha_1 + \alpha)\right)^{\geq t'_{d+1}} \\ = & \ (B_d - L_d)^{\geq t'_{d+1}} - 2m'\beta + (m' - 1)\alpha - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}} \\ = & \ \left((B_d - L_d)^{\geq t_{d+1}} - (B_d - L_d)^{\left[t_{d+1}, t'_{d+1}\right)}\right) - 2m'\beta + (m' - 1)\alpha \\ & \ - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}} \\ = & \ H_d - (B_d - L_d)^{\left[t_{d+1}, t'_{d+1}\right)} - 2m'\beta + (m' - 1)\alpha - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}}. \end{aligned}$$

Observe that each element in $B_d - L_d$ has an even count, B_d contains at least m' copies of α_1 (since $m'A \subseteq B_d$), and L_d contains exactly one copy of α_1, therefore $B_d - L_d - (m' - 1)\alpha_1$ is a valid collection in which each element has an even count (recall that m' is odd). Denote $C = \frac{1}{2}\left((B_d - L_d - (m' - 1)\alpha_1)^{\left[t_{d+1}, t'_{d+1}\right)}\right) = \frac{1}{2}\left((B_d - L_d)^{\left[t_{d+1}, t'_{d+1}\right)} - (m' - 1)\alpha_1^{< t'_{d+1}}\right)$. Next, we can write

$$\begin{aligned}
H'_d = &\; H_d - (B_d - L_d)^{[t_{d+1}, t'_{d+1})} - 2m'\beta + (m'-1)\alpha - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}} \\
= &\; H_d - 2C - (m'-1)\alpha_1^{<t'_{d+1}} - 2m'\beta + (m'-1)\alpha - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}} \\
= &\; H_d - 2C - (m'-1)\left(\alpha_1 - \alpha_1^{\geq t'_{d+1}}\right) - 2m'\beta + (m'-1)\alpha - m'A^{\geq t'_{d+1}} + \alpha_1{}^{\geq t'_{d+1}} \\
= &\; H_d - 2C - (m'-1)\alpha_1 - 2m'\beta + (m'-1)\alpha - m'(A-\alpha_1)^{\geq t'_{d+1}}.
\end{aligned}$$

Since $m' \geq 1$ (being an odd nonnegative integer), we get that $|H'_d| = |H_d| - 2|C| - 2m' - m'\left|(A-\alpha_1)^{\geq t'_{d+1}}\right| < |H_d| - 2|C|$. Therefore, $|L'_d| + \frac{|H'_d|}{2} < |L_d| + \frac{|H_d|}{2} + |C|$, and condition (c) of Claim 18 holds with respect to B, B', C, and $i = d$. In addition, $B_{d+1} \overset{\text{Obs.4}}{=} \frac{1}{2^{d+1}} B^{<t_{d+1}}$, and in particular the top values of all elements in B_{d+1} are lower than t_{d+1}. Since the top values of all elements in C are at least t_{d+1}, we have that $B_{d+1} \cap C = \emptyset$, and condition (d) of Claim 18 holds with respect to B, B', C, and $i = d$. From Claim 9, there is an integer $x > 0$ and a convexed l-collection $\hat{A}$ such that $|\hat{A}| < |A|$ and $(A-\alpha_1)^{<t'_{d+1}} = 2^x \hat{A}$, and so

$$\begin{aligned}
H'_d = &\; H_d - 2C - (m'-1)\alpha_1 - 2m'\beta + (m'-1)\alpha - m'(A-\alpha_1)^{\geq t'_{d+1}} \\
= &\; H_d - 2C - (m'-1)\alpha_1 - 2m'\beta + (m'-1)\alpha - m'\left((A-\alpha_1) - (A-\alpha_1)^{<t'_{d+1}}\right) \\
= &\; H_d - 2C - 2m'\beta + (m'-1)\alpha - (m'A - \alpha_1) + m'2^x\hat{A},
\end{aligned}$$

and

$$\begin{aligned}
B'_{d+1} = &\; \tfrac{1}{2}(B'_d - L'_d - H'_d) \\
= &\; \tfrac{1}{2}((B_d - m'(2\beta + A) + m'\alpha) - (L_d - \alpha_1 + \alpha) \\
&\; -(H_d - 2C - 2m'\beta + (m'-1)\alpha - (m'A - \alpha_1) + m'2^x\hat{A})) \\
= &\; \tfrac{1}{2}(B_d - L_d - H_d) + C - m'2^{x-1}\hat{A} \\
= &\; B_{d+1} + C - m'2^{x-1}\hat{A}
\end{aligned}$$

Since $x > 0$, $m'2^{x-1}$ is an integer. Therefore, requirement (e) in Claim 18 holds with respect to $B, B', C, \hat{A}, i = d$, and $m'' = m'2^{x-1}$, and the claim follows.

□

Claim 20. *Let $B' = B + m\varepsilon$ be a type II elementary folding. For $d = d_m$, we have*

1. $\vec{s}'_{d-1} = \vec{s}_{d-1}$,

2. $s'_d = s_d + 1,$

3. $r' = d + 1.$

Proof. Since the folding is elementary, $\varepsilon \notin B$, and for every $i \geq 0$ we have $t_i > 0$. Therefore, $\varepsilon^{<t_i} = \varepsilon$ and $\varepsilon^{[t_i+1,t_i)} = \emptyset$. From Claim 16, we get that

$$\begin{aligned} B'_d = &\ B_d + \tfrac{m}{2^d}\varepsilon, \\ \vec{L}'_{d-1} = &\ \vec{L}_{d-1}, \\ \vec{H}'_{d-1} = &\ \vec{H}_{d-1}. \end{aligned}$$

.

From $\vec{L}'_{d-1} = \vec{L}_{d-1}$ and $\vec{H}'_{d-1} = \vec{H}_{d-1}$, it follows that $\vec{s}'_{d-1} = \vec{s}_{d-1}$. Since $\frac{m}{2^d}$ is an odd integer (by definition) and $\varepsilon \notin L_d \subseteq B$, $L'_d = \text{mod2}\left(B'_d\right) = \text{mod2}\left(B_d + \frac{m}{2^d}\varepsilon\right) \overset{\text{Obs.2}}{=} \text{mod2}\left(B_d + \varepsilon\right) \overset{\text{Obs.2}}{=} L_d + \varepsilon - 2(L_d \cap \varepsilon) = L_d + \varepsilon$. If $d = 0$ then $s'_d = |L'_d| = |L_d| + 1 = s_d + 1$, and otherwise $s'_d = |L'_d| - |L'_{d-1}| - \frac{|H'_{d-1}|}{2} + \max(s'_{d-1}, 0) = |L_d| + 1 - |L_{d-1}| - \frac{|H_{d-1}|}{2} + \max(s_{d-1}, 0) = s_d + 1$. Finally, as $\varepsilon \in L'_d$ (and in particular $r' > d$), it follows that $t'_{d+1} = min^t(L'_t) = top(\varepsilon) = 0$, $B'_{d+1} \overset{\text{Obs.4}}{=} \frac{1}{2^{d+1}} B^{<0} = \emptyset$, and so $r' = d + 1$. □

Finally, we prove the main claim in this section.

Proof of Claim 14. The correctness of the claim follows immediately from Claims 15, 19, and 20. □

C.2.3 The FOLD Procedure

Using the notation and definitions given in the previous sections, we now give an explicit description of the FOLD procedure.

In general, it is easy to assert that when the precondition holds, the returned collection B' from the RIGHT-FOLD procedure is a folding of the input collection B, where each one of the 2^g copies of α in B' is obtained by concatenating all elements in A and two copies of β. Since a right-folding adds to the collection 2^g copies of α while reducing 2^g repeats of the collection $2\beta + A$ of size $2 + 2^{r-g} - 1 = 1 + 2^{r-g}$, the size of the folded collection B' has decreased by 2^r with respect to the size of the original collection B.

Right-folding Properties

In this section we show certain characteristics of right-foldings.

Claim 21. *There is a right folding of a collection B if and only if $s_r < 0$.*

Proof. For the first direction of the proof, assume that there is a right folding B' of B. From Claim 10, $|L_g| \geq \max(s_g, 0)$. Since $H_g \neq \emptyset$, we get that $-|L_g| - \frac{|H_g|}{2} + \max(s_g, 0) < 0$. This, in turn, implies that $s_{g+1} = |L_{g+1}| - |L_g| - \frac{|H_g|}{2} + \max(s_g, 0) < |L_{g+1}|$. If $r = g+1$, then $s_r = s_{g+1} < |L_{g+1}| = 0$, and the claim follows. Otherwise, $L_{g+1} \neq \emptyset$, and in particular $-|L_{g+1}| - \frac{|H_{g+1}|}{2} + \max(s_{g+1}, 0) < 0$. Inductively, this shows that $s_r < 0$.

For the second direction, assume that $s_r < 0$. Assume that for some $i < r$ we have that $-|L_i| - \frac{|H_i|}{2} + \max(s_i, 0) < 0$, and that $H_d = \emptyset$ and $L_d \neq \emptyset$ for all $i < d < r$. Note that this requirement holds for $i = r-1$, since $-|L_{r-1}| - \frac{|H_{r-1}|}{2} + \max(s_{r-1}, 0) = |L_r| - |L_{r-1}| - \frac{|H_{r-1}|}{2} + \max(s_{r-1}, 0) = s_r < 0$, and there are no integers d such that $r-1 < d < r$. If $H_i \neq \emptyset$, then also $L_i \neq \emptyset$ (since $L_i = \emptyset$ implies that $min^t(L_i) = \infty$, and by definition $H_i = (B_i - L_i)^{\geq\infty} = \emptyset$), and therefore the requirements for the existence of a right-folding hold for $g = i$. Else, $H_i = \emptyset$, and so $-|L_i| + \max(s_i, 0) < 0$. This implies that $L_i \neq \emptyset$ and that $i \neq 0$ (as $|L_0| = s_0$), and so $|L_i| > \max(s_i, 0) \geq s_i = |L_i| - |L_{i-1}| - \frac{H_{i-1}}{2} + \max(s_{i-1}, 0)$, and we get that $-|L_{i-1}| - \frac{H_{i-1}}{2} + \max(s_{i-1}, 0) < 0$. Inductively, for some $i' < r$, it must hold that $H_{i'} \neq \emptyset$, $H_d = \emptyset$ for every $i' < d < r$, and $L_d \neq \emptyset$ for every $i' \leq d < r$, meeting the requirements for the existence of a right folding for $g = i'$. $\square$

Throughout the remaining of this section, assume that B is a collection satisfying the pre-condition in Procedure RIGHT-FOLD, and let $B' = B - 2^g(2\beta + A) + 2^g\alpha$ be the output of the procedure (where g, r, β, $A = \{\alpha_1, \ldots, 2^{r-g-1}\alpha_{r-g}\}$, and α are as defined in the procedure). Note that when $\alpha \notin B$, B' is also a type I elementary folding of B. We later show that all right-foldings preformed in line 7 of the FOLD procedure are elementary.

Claim 22. *If B' is an elementary folding of B, then*

1. $\vec{L}'_{g-1} = \vec{L}_{g-1}$*, and* $L'_g = L_g - \alpha_1 + \alpha$.
2. $\vec{H}'_{g-1} = \vec{H}_{g-1}$*, and* $H'_g = H_g - 2\beta$.

3. *For every $g < i < r$, $L'_i = L_i - \alpha_{i-g}$, and $H'_i = H_i = \emptyset$.*

4. *$B'_r = B_r = \emptyset$ (thus $r' \leq r$).*

5. $|B'| = |B| - 2^r$.

Proof. We start by showing the first two items in the claim. Since $\beta \in H_g \overset{\text{Obs.4}}{\subseteq} B^{<t_g}$, it follows that for every $\alpha_j \in A$ and every $0 \leq i \leq g$, $top\left(\alpha_j\right) \leq top\left(\alpha\right) = top\left(\beta\right) < t_g \leq t_i$. Therefore, from Claim 17 and the fact that $d_{2^g} = g$, we get that $\vec{L}'_{g-1} = \vec{L}_{g-1}$ (and in particular $\vec{t}'_g = \vec{t}_g$), $\vec{H}'_{g-1} = \vec{H}_{g-1}$, and $B'_g = B_g - (2\beta + A) + \alpha$. Since $\text{mod2}\left(2\beta + A\right) = \alpha_1 \in L_g$, we have $\text{mod2}\left(B_g - (2\beta + A)\right) \overset{\text{Obs.2}}{=} \text{mod2}\left(B_g + \alpha_1\right) \overset{\text{Obs.2}}{=} \text{mod2}\left(B_g\right) + \alpha_1 - 2(\text{mod2}\left(B_g\right) \cap \alpha_1) = L_g + \alpha_1 - 2(L_g \cap \alpha_1) = L_g - \alpha_1$. As $\alpha \notin L_g + \alpha_1$ (follows from $\alpha \notin B$), we get that $L'_g = \text{mod2}\left(B'_g\right) = \text{mod2}\left(B_g - (2\beta + A) + \alpha\right) \overset{\text{Obs.2}}{=} \text{mod2}\left((L_g - \alpha_1) + \alpha\right) = L_g - \alpha_1 + \alpha$. By definition, α_1 is a minimal element in L_g, therefore $top\left(\alpha_1\right) = min^t\left(L_g\right)$, and so $top\left(\alpha_1\right) \leq min^t\left(L_g - \alpha_1\right)$. In addition, $top\left(\alpha_1\right) \leq top\left(\alpha\right)$, and therefore $top\left(\alpha_1\right) \leq \min(min^t\left(L_g - \alpha_1\right), top\left(\alpha\right)) = min^t\left(L_g - \alpha_1 + \alpha\right) = min^t\left(L'_g\right)$. On the other hand, $top\left(\beta\right) = top\left(\alpha\right)$, and therefore $min^t\left(L'_g\right) = min^t\left(L_g - \alpha_1 + \alpha\right) \leq top\left(\beta\right)$. From the minimality of β in H_g, $H_g = H_g^{\geq top(\beta)} = ((B_g - L_g)^{\geq min^t(L_g)})^{\geq top(\beta)} = ((B_g - L_g)^{\geq top(\alpha_1)})^{\geq top(\beta)} = (B_g - L_g)^{\geq \max(top(\alpha_1), top(\beta))} = (B_g - L_g)^{\geq top(\beta)}$. Since $top\left(\alpha_1\right) \leq min^t\left(L'_g\right) \leq top\left(\beta\right)$, we get that $H_g = (B_g - L_g)^{\geq top(\beta)} \subseteq (B_g - L_g)^{\geq min^t(L'_g)} \subseteq (B_g - L_g)^{\geq top(\alpha_1)} = H_g$, and so $(B_g - L_g)^{\geq min^t(L'_g)} = H_g$. In addition, $\beta^{\geq min^t(L'_g)} = \beta$, and $(A - \alpha_1)^{\geq min^t(L'_g)} = \emptyset$ (since for all $i > 0$, $top\left(\alpha_i\right) < top\left(\alpha_1\right) \leq min^t\left(L'_g\right)$). Thus, we get that $H'_g = (B'_g - L'_g)^{\geq min^t(L'_g)} = ((B_g - 2\beta - A + \alpha) - (L_g - \alpha_1 + \alpha))^{\geq min^t(L'_g)} = (B_g - L_g)^{\geq min^t(L'_g)} - 2\beta^{\geq min^t(L'_g)} - (A - \alpha_1)^{\geq min^t(L'_g)} = H_g - 2\beta$, as required.

Next, we turn to show item 3 in the claim, which is relevant only for the case where $g < r - 1$. We prove this item inductively for all $g < i < r$. Note that $B'_{g+1} = \frac{1}{2}(B'_g - L'_g - H'_g) = \frac{1}{2}((B_g - 2\beta - A + \alpha) - (L_g - \alpha_1 + \alpha) - (H_g - 2\beta)) = \frac{1}{2}((B_g - L_g - H_g) - (A - \alpha_1)) = B_{g+1} - \frac{1}{2}A$. Now, assume that for some $g < i < r$, $B'_i = B_i - \frac{1}{2^{i-g}}A$. Note that $\alpha_{i-g} \in L_i$, and in particular $L_i \cap \alpha_{i-g} = \alpha_{i-g}$. Therefore, $L'_i = \text{mod2}\left(B'_i\right) = \text{mod2}\left(B_i - \frac{1}{2^{i-g}}A\right) \overset{\text{Obs.2}}{=} \text{mod2}\left(B_i\right) + \text{mod2}\left(\frac{1}{2^{i-g}}A\right) - 2(\text{mod2}\left(B_i\right) \cap \text{mod2}\left(\frac{1}{2^{i-g}}A\right)) = L_i + \alpha_{i-g} - 2(L_i \cap \alpha_{i-g}) = L_i + \alpha_{i-g} - 2\alpha_{i-g} = L_i - \alpha_{i-g}$, as required. In addition, since $min^t\left(L'_i\right) = min^t\left(L_i - \alpha_{i-g}\right) \geq min^t\left(L_i\right) = top\left(\alpha_{i-g}\right)$, we get that $H'_i = (B'_i - L'_i)^{\geq min^t(L'_i)} = ((B_i - \frac{1}{2^{i-g}}A) - (L_i - \alpha_{i-g}))^{\geq min^t(L'_i)} \subseteq (B_i - L_i - (\frac{1}{2^{i-g}}A - \alpha_{i-g}))^{\geq min^t(L_i)} \subseteq$

$H_i = \emptyset$, and so $H'_i = \emptyset$, as required. Finally, it follows that $B'_{i+1} = \frac{1}{2}(B'_i - L'_i - H'_i) = \frac{1}{2}((B_i - \frac{1}{2^{i-g}}A) - (L_i - \alpha_{i-g}) - H_i) = B_{i+1} - \frac{1}{2}(\frac{1}{2^{i-g}}A - \alpha_{i-g}) = B_{i+1} - \frac{1}{2^{i+1-g}}A$, as required for the next inductive step.

Item 4 in the claim follows from the fact that $B'_r = B_r - \frac{1}{2^{r-g}}A = \emptyset - \emptyset = \emptyset$, and item 5 is obtained from the fact that $|B'| = |B| - 2^g(2+|A|) + 2^g = |B| - 2^g(2+2^{r-g} - 1) + 2^g = |B| - 2^r$. □

Claim 23. *If B' is an elementary folding of B, then $\vec{s'}_{r-1} = \vec{s}_{r-1}$ and $s'_r = s_r + 1$. In addition, $r' \leq r$, where if $s'_r < 0$ then $r' = r$.*

Proof. By definition 7, the values in the series $\vec{s'}$ depend only on sizes of collections in $\vec{L'}$ and $\vec{H'}$. These sub-collection sizes may be inferred from Claim 22, and their assignments in definition 7 imply the correctness of the claim in a straightforward manner. □

Let β be a palindrome obtained by concatenating zero or more l-blocks. If β is obtained by concatenating an odd number of blocks, β is of the form $\beta = \beta_1\beta_2\ldots\beta_{q-1}\beta_q\beta_{q-1}\ldots\beta_2\beta_1$ (where each β_i is an l-block), whereas if β is obtained by concatenating an even number of blocks it is of the form $\beta = \beta_1\beta_2\ldots\beta_{q-1}\beta_q\varepsilon\beta_q\beta_{q-1}\ldots\beta_2\beta_1$. Call β_q or ε respectively the *center* of β, in these two cases. Note that a center of an l-block β is β.

Definition 9. *Say that an l-BFB palindrome collection B has* unique centers *if all elements in collections of the form H_d are l-blocks, and for every $\beta \in L_d$ and $\beta' \in L'_d$ (for some possibly equal integers d and d') such that $\beta \neq \beta'$, the centers of β and β' differ.*

Claim 24. *If B has unique centers then B' is an elementary folding of B, and B' has unique centers.*

Proof. To prove the folding is elementary, we need to show that $\alpha \notin B$. Note that $\beta \in H_g$, $\alpha_1 \in L_g$, $top(\alpha) = top(\beta)$, and the center of α is the the center of α_1. Assume by contradiction that $\alpha \in B$. Since $top(\alpha) = top(\beta)$, Observation 4 implies that $\alpha \in L_g + H_g$. Since α is not an l-block, $\alpha \notin H_g$. Since α_1 and α have the same center, and since $\alpha_1 \in L_g$ and B has unique centers, it follows that $\alpha \notin L_g$, leading to a contradiction.

The fact that B' has unique centers follows from the contents of collections in the series $\vec{L'}$ and $\vec{H'}$, as given in Claim 22. □

Claim 25. *Let B be an l-BFB palindrome collection with unique centers. Then, for $i = -s_r - \min(s_{r-1}, 0)$, it is possible to produce a series of collections $B^0 = B, B^1, B^2, \ldots, B^i$, each collection B^j obtained by applying an elementary right-foldings over the preceding collection B^{j-1}. In addition, for every $0 \leq j \leq -s_r$,*

- $|B^j| = |B| - 2^r j$,
- $\vec{s}^{\,j}_{r-1} = \vec{s}_{r-1}$,
- $s^j_r = s_r + j$,

and for each $-s_r \leq j \leq -s_r - \min(s_{r-1}, 0)$,

- $r^j \leq r - 1$,
- $|B^j| = |B| + 2^{r-1}(s_r - j)$,
- $\vec{s}^{\,j}_{r-2} = \vec{s}_{r-2}$,
- $s^j_{r-1} = s_{r-1} + j + s_r$.

Proof. From Claim 11, $s_r \leq 0$. Note that $s_r = -|L_{r-1}| - \frac{|H_{r-1}|}{2} + \max(s_{r-1}, 0) \leq \max(s_{r-1}, 0)$. When $s_r = 0$, $\max(s_{r-1}, 0) \geq 0$, and so $\min(s_{r-1}, 0) = 0$ and in particular $-s_r - \min(s_{r-1}, 0) = 0$ and the claim holds trivially. Otherwise, $s_r < 0$, thus $-s_r - \min(s_{r-1}, 0) > 0$, and we continue to assert the correctness of the claim.

In the remaining case, $s_r < 0$, and from Claims 21, 24, and 23 there is an elementary right-folding B^1 of $B^0 = B$ with unique centers, such that $\vec{s}^{\,1}_{r-1} = \vec{s}_{r-1}$ and $s^1_r = s_r + 1$. Note that $r^1 \leq r$, where $s^1_r < 0$ implies that $r_1 = r$. Similarly, it is possible to apply a series of a total amount of $x = -s_r$ right-foldings $B = B^0, B^1, \ldots, B^x$, where for every $j < x$ we have that $r_j = r$ and $r_x \leq r$, and for every $j \leq x$ we have that $\vec{s}^{\,j}_{r-1} = \vec{s}_{r-1}$ and $s^j_r = s_r + j$. Since each such a right-folding decreases the size of the collection by 2^r elements, $|B^j| = |B| - 2^r j$, hence the first part of the claim.

After performing $x = -s_r$ right-foldings, we get the collection B^x for which $r_x \leq r$, $\vec{s}^{\,j}_{r-1} = \vec{s}_{r-1}$, $s^j_r = s_r + x = 0$, and $|B^x| = |B| - 2^{r+1}x = |B| + 2^{r+1}s_r$. If $-\min(s_{r-1}, 0) = 0$, then the second part of the claim follows immediately. Else, $-\min(s_{r-1}, 0) = -\min(s^x_{r-1}, 0) > 0$, thus $s^x_{r-1} = s_{r-1} < 0$, and $\max(s^x_{r-1}, 0) = 0$. Since $0 = s^x_r = -|L^x_{r-1}| -$

$\frac{|H_{r^x-1}|}{2} + \max(s^x_{r-1}, 0) = -|L^x_{r-1}| - \frac{|H_{r^x-1}|}{2}$, it follows that $L^x_{r-1} = H^x_{r-1} = \emptyset$, and therefore $r^x \leq r-1$. On the other hand, from Claim 11 and the fact that $s^x_{r-1} \neq 0$ we get that $r^x \geq r-1$, and thus $r^x = r-1$. As above, it is possible to apply additional consecutive $y = -s^x_{r-1} = -s_{r-1} = -\min(s_{r-1}, 0)$ right-foldings, where each such folding maintains the signature values at positions 0 to $r-2$, increases by 1 the signature value at position $r-1$ with respect to the preceding collection in the series, and decreases the collection size by 2^{r-1}. Hence, for $-s_r \leq j \leq -s_r - \min(s_{r-1}, 0)$, we have that $r^j \leq r-1$, $|B^j| = |B^x| - 2^{r-1}(j-x) = |B| + 2^r s_r - 2^{r-1}(j+s_r) = |B| + 2^{r-1}(s_r - j)$, $\vec{s}^{\,j}_{r-2} = \vec{s}_{r-2}$, and $s^j_{r-1} = s^x_{r-1} + j - x = s_{r-1} + j + s_r$, as required. $\square$

Correctness of the FOLD Procedure

Throughout this section, B and n correspond to an l-block collection and an integer given as an input to the FOLD procedure. When $n \neq |B|$, denote $d' = d_{|B|-n}$.

Claim 26. *If there is a folding B' of B of size $n \neq |B|$, then there is an integer $0 \leq d \leq d'$ such that $\vec{s}'_{d-1} = \vec{s}_{d-1}$, $s'_d \geq s_d + 1$, and $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$. In addition, if $d < d'$, then $s'_d \geq s_d + 2$.*

Proof. Assume there is a folding B' of B of size $n \neq |B|$, and let $d \geq 0$ be the integer such that $\vec{s}_{d-1} = \vec{s}'_{d-1}$ and $s'_d \geq s_d + 1$ (whose existence is implied by Claim 14). Since $\vec{s}_{d-1} = \vec{s}'_{d-1}$, it follows that $\Delta_d = \Delta'_d$. Thus, $n = |B'| \overset{\text{Clm.12}}{=} 2^d(|B'_d| + |L'_d| - s'_d) + \Delta'_d \overset{\text{Clm.12}}{\geq} 2^d \max(s'_d, 0) + \Delta'_d \geq 2^d \max(s_d + 1, 0) + \Delta_d$. In addition, $|B| = 2^d(|B_d| + |L_d| - s_d) + \Delta_d$, therefore $|B| - n = 2^d\left(|B_d| + |L_d| - s_d - |B'_d| - |L'_d| + s'_d\right)$. Since all parameters in the right-hand side of the latter equation are integers, $|B| - n$ divides by 2^d, and in particular $d \leq d'$. Furthermore, if $d < d'$, then $\frac{|B|-n}{2^d}$ is even, and therefore $|B_d| + |L_d| - s_d - |B'_d| - |L'_d| + s'_d$ is also even. As $|B_d| + |L_d|$ is even, as well as $|B'_d| + |L'_d|$, it follows that $s'_d - s_d$ has to be even. Since $s'_d > s_d$, it follows that $s'_d \geq s_d + 2$. $\square$

Claim 27. *If there is a folding of B of size n then FOLD(B, n) returns such a folding B', and otherwise FOLD(B, n) returns "FAILED". In addition, if $n \neq |B|$ and FOLD(B, n) has returned B', then for the maximum integer $0 \leq d \leq d'$ for which $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$ (whose existence is guaranteed by Claim 26), $\vec{s}'_{d-1} = \vec{s}_{d-1}$ and $r' \leq d+1$, and if $r' = d+1$ then $s'_d = s_d + 1$ in case $d = d'$ and $s'_d = s_d + 2$ in case $d < d'$.*

Proof. When there is no folding of B of size n, then in particular $n \neq |B|$, and the procedure does not halt at line 1. In addition, from Claim 26, the condition in line 4 does not met, and the procedure returns "FAILED" in line 10 as required.

Else, there is a folding of B of size n, and we show that the procedure finds such a folding sustaining the stated requirements. When $|B| <= n$, the FOLD procedure halts by returning $B + (n - |B|)\varepsilon$ in line 1, which is in particular a folding of B of size n as required. In addition, if $|B| < n$, we have from Claim 20 that $\vec{s}'_{d-1} = \vec{s}_{d-1}$, $s'_d = s_d + 1$, and $r' = d + 1$, thus the remaining requirements in the claim hold. Otherwise, $n < |B|$, and from Claim 26 the condition in line 4 holds, therefore in line 5 of the FOLD procedure, the value of the parameter d is selected to be the maximum integer in the range $0 \leq d \leq d'$ such that $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$.

Let $B^0 = B + 2^d\varepsilon$ be the value of the collection B' after executing line 5. Thus $|B^0| = |B| + 2^d$, and from Claim 20, we have that

1. $\vec{s}^{\,0}_{d-1} = \vec{s}_{d-1}$,
2. $s^0_d = s_d + 1$,
3. $r^0 = d + 1$.

From the proof of Claim 20 and the fact that B is an l-block collection it can be seen that B^0 has unique centers. From Conclusion 1, $s^0_{d+1} = \frac{\Delta^0_{d+1} - |B^0|}{2^{d+1}} = \frac{\Delta_d + 2^d \text{abs}(s_d+1) - |B| - 2^d}{2^{d+1}}$. From Claim 25, the collection B^0 can undergo a series of i right-foldings producing the sequence $B^0, B^1, \ldots, B^i$, where $i = -s^0_{d+1} - \min(s^0_d, 0)$. The size of the collection B^i according to Claim 25 is $|B^i| = |B^0| + 2^d(s^0_{d+1} - i) = (|B| + 2^d) + 2^d(2s^0_{d+1} + \min(s^0_d, 0)) = |B| + 2^d + 2^d\left(\frac{\Delta_d + 2^d \text{abs}(s_d+1) - |B| - 2^d}{2^d} + \min(s_d + 1, 0)\right) = \Delta_d + 2^d(\text{abs}(s_d + 1) + \min(s_d + 1, 0)) = \Delta_d + 2^d \max(s_d + 1, 0)$. From the condition in line 4, $n \geq 2^d \max(s_d + 1, 0) + \Delta_d = |B^i|$, and in particular there exists some $0 \leq j \leq i$ such that $|B^j| \leq n$. The sequence of right-foldings computed by the loop lines 6-7 is a prefix of such a right-folding sequence (i.e. after x iterations of the loop, $B' = B^x$), where the loop terminates after j iterations for a minimal integer j such that $|B^j| \leq n$. After executing line 8, B' is a folding of B of size $|B'| = |B^j| + (n - |B^j|) = n$, and so the output B' of the procedure is a folding of B of size n, as required.

To complete the proof, we need to show that when $n < |B|$, $\vec{s}'_{d-1} = \vec{s}_{d-1}$ and $r' \leq d+1$, and if $r' = d+1$ then $s'_d = s_d + 1$ in case $d = d'$ and $s'_d = s_d + 2$ in case $d < d'$. To do so, we consider two cases for the number of loop iterations j conducted by the procedure. Note that $j > 0$, since in the first iteration we have that $|B^0| = |B| + 2^d > n$.

1. $0 < j \leq -s^0_{d+1}$**.** In this case, Claim 25 and the loop termination condition imply that $n \geq |B^j| = |B^0| - 2^{d+1}j = |B| + 2^d - 2^{d+1}j = |B| + 2^d(1-2j)$, and that $n < |B^{j-1}| = |B| + 2^d(1-2(j-1))$, therefore, $2j - 3 < \frac{|B|-n}{2^d} \leq 2j-1$. Note that when $d = d'$, $\frac{|B|-n}{2^d}$ is odd, hence $\frac{|B|-n}{2^d} = 2j-1$, whereas when $d < d'$, $\frac{|B|-n}{2^d}$ is even, and $\frac{|B|-n}{2^d} = 2j-2$.

In the case where $d = d'$, $|B^j| = |B| + 2^d(1-2j) = |B| - 2^d(\frac{|B|-n}{2^d}) = n$, thus no ε elements are added to the collection in line 8 of the procedure and the returned collection is $B' = B^j$. From Claim 25, $r' = r^0 = d+1$, and $\vec{s}'_d = \vec{s}^0_d$, i.e. $\vec{s}'_{d-1} = \vec{s}^0_{d-1} = \vec{s}_{d-1}$ and $s'_d = s^0_d = s_d + 1$, and the claim follows.

In the case where $d < d'$, $|B^j| = |B| + 2^d(1-2j) = |B| - 2^d(2j-2+1) = |B| - 2^d(\frac{|B|-n}{2^d} + 1) = n - 2^d$, thus after line 8 of the procedure the returned collection is $B' = B^j + 2^d\varepsilon$. It may be asserted that ε is the unique minimal element in B^0_d (as all other elements are l-blocks with higher top values), and thus this element participates in the right-folding that transforms B^0 to B^1. Therefore, for each $1 \leq j' \leq j$, $\varepsilon \notin B^{j'}$, and in particular B' is a type II elementary folding of B^j. From Claim 20, $r' = d+1$, $\vec{s}'_{d-1} = \vec{s}^0_{d-1} = \vec{s}_{d-1}$, and $s'_d = s^0_d + 1 = s_d + 2$, hence the claim follows.

2. $-s^0_{d+1} < j \leq -s^0_{d+1} - \min(s^0_d, 0)$**.** In this case, Claim 25 and the loop termination condition imply that $n \geq |B^j| = |B^0| + 2^d(s^0_{d+1} - j) = (|B| + 2^d) + 2^d(s^0_{d+1} - j) = |B| + 2^d(s^0_{d+1} - j + 1)$, and $n < |B^{j-1}| = |B| + 2^d(s^0_{d+1} - j + 2)$. Therefore, $-s^0_{d+1} + j - 2 < \frac{|B|-n}{2^d} \leq -s^0_{d+1} + j - 1$. Since $\frac{|B|-n}{2^d}$ is an integer, it follows that $\frac{|B|-n}{2^d} = -s^0_{d+1} + j - 1$, therefore $|B^j| = n$, and consequentially after line 8 of the procedure the returned collection is $B' = B^j$. From Claim 25, $r' \leq d$, and $\vec{s}'_{d-1} = \vec{s}^0_{d-1} = \vec{s}_{d-1}$, and the claim follows. □

Finally, we now prove the correctness of the FOLD procedure, as formulated by Claim 28.

Claim 28. *Let B be an l-block collection and let $n \geq 0$ be an integer. FOLD(B,n) returns a folding B' of B of size n if such a folding exists, and otherwise it returns "FAILED".*

In addition, for every l-block collection B^* such that $|B| = |B^*|$ and $\vec{s}(B) \leq \vec{s}(B^*)$, if there is a folding B'^* of B^* of size n, then FOLD(B,n) returns a collection B' such that $\vec{s}(B') \leq \vec{s}(B'^*)$.

Proof. Claim 27 proves the first statement in Claim 28, thus it remains to show that for every l-block collection B^* such that $|B| = |B^*|$ and $\vec{s} \leq \vec{s}^*$, if there is a folding B'^* of B^* of size n, then FOLD(B,n) returns a collection B' such that $\vec{s}' \leq \vec{s}'^*$.

First, note that when $n = |B| = |B^*|$, then in particular B^* and B are minimum signature n-size foldings of B^* and B, respectively (Claim 14), and thus $B \leq^s B'^*$ for every n-size folding B'^* of B^*. Since in this case FOLD(B,n) returns B, the claim follows. Otherwise, $n \neq |B|$, and we first show that FOLD(B,n) returns a folding B' of B of size n if such a folding exists, and otherwise it returns "FAILED".

In the reminder of this proof we assume that $n \neq |B| = |B^*|$, and note that $d' = d_{|B|-n} = d_{|B^*|-n}$. Since $\vec{s} \leq \vec{s}^*$, either $\vec{s} = \vec{s}^*$, or $\vec{s} < \vec{s}^*$ and there is an integer i such that $\vec{s}_{i-1} = \vec{s}^*_{i-1}$ and $s_i < s^*_i$.

We first show that if is a folding B'^* of B^* of size n, FOLD(B,n) returns a folding B' of B of size n satisfying $\vec{s}' \leq \vec{s}'^*$. In this case, Claim 26 states that there is an integer $0 \leq d^* \leq d'$ such that $\vec{s}'^*_{d^*-1} = \vec{s}^*_{d^*-1}$, $s'^*_{d^*} \geq s^*_{d^*} + 1$, and $n \geq 2^{d^*} \max(s^*_{d^*} + 1, 0) + \Delta^*_{d^*}$. Consider two cases: (1) $\vec{s}_{d^*-1} = \vec{s}^*_{d^*-1}$, which occurs when $\vec{s} = \vec{s}^*$ or when $\vec{s} < \vec{s}^*$ and $i \geq d^*$, and (2) $\vec{s}_{d^*-1} < \vec{s}^*_{d^*-1}$, which occurs when $\vec{s} < \vec{s}^*$ and $i < d^*$.

(1) $\vec{s}_{d^*-1} = \vec{s}^*_{d^*-1}$. In this case, $n \geq 2^{d^*} \max(s^*_{d^*} + 1, 0) + \Delta^*_{d^*} \geq 2^{d^*} \max(s_{d^*} + 1, 0) + \Delta_{d^*}$. Thus, when executing FOLD(B,n), the condition in line 4 is met and the algorithm does not return "FAILED". From Claim 27, FOLD(B,n) returns an n-size folding B' of B, such that for the maximum integer $0 \leq d \leq d'$ for which $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$ we have that $\vec{s}'_{d-1} = \vec{s}_{d-1}$ and $r' \leq d+1$, and if $r' = d+1$ then $s'_d = s_d + 1$ in case $d = d'$ and $s'_d = s_d + 2$ in case $d < d'$. By selection, $d^* \leq d \leq d'$. If $d^* < d$, then $\vec{s}'_{d^*} = \vec{s}_{d^*} = \vec{s}^*_{d^*} < \vec{s}'^*_{d^*}$, and in particular $\vec{s}' \leq \vec{s}'^*$ and the claim follows. If $d^* = d$, then $\vec{s}'_{d-1} = \vec{s}_{d-1} = \vec{s}^*_{d-1} = \vec{s}'^*_{d-1}$. If $r' < d+1$ then $\vec{s}' \leq \vec{s}'^*$ from Claim 13, and the claim follows. If $r' = d+1$, then $s'_d = s_d + 1$ in case $d = d'$ and $s'_d = s_d + 2$ in case $d < d'$. In addition, from Claim 26, $s'^*_d \geq s_d + 1$ in case $d = d'$ and $s'^*_d \geq s_d + 2$ in case $d < d'$, thus in both cases $s'_d \leq s'^*_d$. If $s'_d < s'^*_d$ then $\vec{s}'_d < \vec{s}'^*_d$, and in particular $\vec{s}' < \vec{s}'^*$ and the claim follows. If $s'_d = s'^*_d$ then $\vec{s}'_d = \vec{s}'^*_d$, and from Claim 13 $\vec{s}' \leq \vec{s}'^*$ and the claim follows.

(2) $\vec{s}_{d^*-1} < \vec{s}^*_{d^*-1}$. In this case, for $i < d^*$ we have that $\vec{s}_{i-1} = \vec{s}^*_{i-1}$ and $s_i < s^*_i$. Now, $n \geq 2^{d^*}\max(s^*_{d^*}+1,0)+\Delta^*_{d^*} \geq \Delta^*_{d^*} \geq \Delta^*_{i+1} = 2^i\mathrm{abs}(s^*_i)+\Delta^*_i \geq 2^i\max(s^*_i,0)+\Delta^*_i$. Similarly as before, Claims 13 and 27 can be applied to show that $\vec{s}' \leq \vec{s}'^*$.

□

C.2.4 Correctness of Algorithm SEARCH-BFB

Assuming there is a BFB string α^* admitting the algorithm's input count vector $\vec{n}$, the BFB palindrome $\beta^* = \alpha^*\bar{\alpha}^*$ admits the count vector $2\vec{n}$. Let $B^{*k+1} = \emptyset$, B^{*k}, $B^{*k-1},\ldots,B^{*1}$ be the block collection series corresponding to the layers of β^* as described in Chapter 4. Since $B^{k+1} = B^{*k+1} = \emptyset$ (B^{k+1} is initialized in line 1 of Algorithm SEARCH-BFB), we have that $\vec{s}(B^{k+1}) = \vec{s}(B^{*k+1})$. Assume that for some $0 \leq l \leq k$ we have that $\vec{s}(B^{l+1}) \leq \vec{s}(B^{*l+1})$. Recall that the collection B^{*l} is obtained by the wrapping of some folding B'^* of size n_l of B^{*l+1}. Since the wrapping operation does not change element multiplicities and top values, it follows that $\vec{s}(B^{*l}) = \vec{s}(B'^*)$. From Claim 28, the application of the FOLD procedure in the l-th iteration of the loop in lines 2-4 of the algorithm returns a folding B' of B^{l+1} of size n_l, where $\vec{s}(B^l) = \vec{s}(B') \leq \vec{s}(B'^*) = \vec{s}(B^{*l})$. Inductively, the algorithm does not return "FAILED" in each one of the loop iterations, and after the last iteration $\vec{s}(B^1) \leq \vec{s}(B^{*1})$. From the same arguments as above and since B^{*1} can be folded into the single palindrome β^*, it follows that the application of FOLD in line 4 of the algorithm does not fail, and returns a collection containing a single palindrome $\beta = \alpha\bar{\alpha}$, where α is a BFB string admitting $\vec{n}(\alpha) = \vec{n}$.

For the other direction of the proof, assume that the BFB algorithm returned the string α. In this case, the series of collections $B^{k+1}, B^k, \ldots, B^1$ satisfies that each collection B^l is an l-block collection of size n_l and is obtained by folding and wrapping of the preceding collection in the series B^{l+1}. The final collection B^1 is folded into a single BFB palindrome $\beta = \alpha\bar{\alpha}$ admitting the count vector $2\vec{n}$, and therefore α is a BFB string admitting $\vec{n}$.

C.2.5 Time Complexity of Algorithm SEARCH-BFB

Object Representation

The algorithm handles two types of objects: BFB palindromes, and BFB palindrome collections. BFB palindromes are further divided into three subtypes, who are implemented separately: empty palindromes, l-blocks, and composite l-BFB palindromes of the form $\beta\gamma\beta$ (see Claim 1 in Chapter 4). Each BFB palindrome object contains a filed maintaining the top value of the represented palindrome, allowing $O(1)$ time queries of this value. An empty palindrome is represented by an object containing only the top value field (which always holds the value 0), and generating new such objects take $O(1)$ time. An l-block is implemented as an object containing, in addition to the top-value field, a pointer to its internal $(l+1)$-BFB palindrome. Given a pointer to the internal $(l+1)$-BFB palindrome, generating new l-block objects take $O(1)$ time by copying the pointer, and setting the top value field to the top value of the pointed $(l+1)$-BFB palindrome. A composite l-BFB palindrome $\beta\gamma\beta$ is implemented by specifying a pointer to the l-BFB palindrome β, and a list of l-BFB palindromes $\alpha_1, \alpha_2, \ldots, \alpha_p$ representing the convexed l-collection A such that $\gamma = \gamma_A$. Composite palindromes can be generated in a time proportional to the order of their internal convexed l-collection (where the top value field is set to be the top value of β).

A collection $B = \{m_1\beta_1, m_2\beta_2, \ldots, m_q\beta_q\}$ is implemented by an object containing a field which maintains the size $|B|$ of the collection, and two doubly linked lists maintaining the prefixes $\vec{L}_{r-1}$ and $\vec{H}_{r-1}$ of the series $\vec{L}$ and $\vec{H}$ in the decomposition of B, where $r = r(B)$. Note that for $i \geq r$, $L_i = H_i = \emptyset$. Each element L_i or H_i is implemented as a linked list of l-BFB palindromes ordered with nondecreasing top values (it is possible that an H_i list contains multiple repeats of identical elements). Thus, computing $min^t(L_i)$ or $min^t(H_i)$ and extracting minimal elements from such lists is done in $O(1)$ time. Generating an empty collection is done in $O(1)$ time (where the two lists L_{r-1} and H_{r-1} are empty), and duplicating or wrapping a collection B take at most $O(|B|)$ time (note that $r - 1 \leq \log|B|$, since an element $\beta \in B_{r-1}$ corresponds to 2^{r-1} repeats of β in B, and that the total number of elements in all lists L_i and H_i is at most $|B|$).

Type II Elementary Folding

Using the object representation described above, for a collection B such that $\varepsilon \notin B$ and an integer $m > 0$, it is possible to compute a type II elementary folding $B' = B + m\varepsilon$ in $O(|B| + m)$ time as follows. First, the number $d = d_m$ is computed. Note that $d \leq \log m$ (d can be defined as the index of the least significant bit different from 0 in the binary representation of m), and may be computed in $O(\log m)$ time. B' is initialized by copying B, i.e. generating the list $\vec{L}'_{r-1}$ and $\vec{H}'_{r-1}$ (in $O(|B|)$ time). Then, if $d \geq r$, empty collections L'_i and H'_i are added to the prefixes of $\vec{L}'$ and $\vec{H}'$ for $r \leq i \leq d$, and a single ε element is added to L'_d. Else, $d < r$, and a single ε element is added as the first element in L'_d (being of minimum top value among all elements in the list), and elements from collections L'_i and H'_i for $i > d$ are moved into H'_d. This latter modification is performed by first merging each L'_i and H'_i lists for $i > d$ to a single list ordered with nondecreasing top values (in a linear time with respect to the number of elements in the two lists), and then, with increasing index i, each merged list is added to the beginning of H'_d, where 2^{i-d} repeats of each element in the merged list of L'_i and H'_i are added to H'_d. In both cases where $d \geq r$ or $d < r$, it is possible to assert the modification updates properly the representation of B' to represent the collection $B + 2^d\varepsilon$, that $r(B') = d + 1$, and that total time required for the modification is at most $O(|B| + d) = O(|B| + \log m)$. Finally, additional $\frac{m}{2^d} - 1$ repeats of ε are added to H'_d in $O(m)$ time, where now it is possible to assert that B' properly represents the collection $B + m\varepsilon$, and that the total computation time is $O(|B| + m)$.

Right-folding

In order to right-fold a collection B, the algorithm first gets pointers to the elements L_{r-1} and H_{r-1}, in $O(r)$ time for $r = r(B)$. Then, it starts traversing these lists backward for $i = r - 1$ down to g, inclusive, where g is the first encountered index such that $H_g \neq \emptyset$. For each such i, the algorithm extracts the first (minimal) element in the list L_i, and accumulates these elements in a list A. Finally, the algorithm extracts two copies of the minimal element β in H_g, and uses β and A to construct the BFB palindrome $\alpha = \beta\gamma_A\beta$. Then, α is inserted into L_g. As this procedure takes $O(r)$ time and decreases the size of the collection by 2^r, any valid consecutive application of right-foldings over

B takes at most $O(|B|)$ time.

The FOLD Procedure

Consider the application of the FOLD procedure on a collection $B = \{m_1\beta_1, m_2\beta_2, \ldots, m_q\beta_q\}$ and an integer $n \geq 0$. If $n \geq |B|$, the procedure applies in line 1 a type II elementary folding in $O(|B|+n)$ time (Section C.2.5) and hults. Otherwise, given the series $\vec{L}_{r-1}$ and $\vec{H}_{r-1}$, it is possible to compute $\vec{s}_r$ and $\vec{\Delta}_{r+1}$ in $O(r) = O(\log(|B|))$ time. Note that $s_i = 0$ for $i > r$, and $\Delta_i = \Delta_{r+1}$ for $i > r+1$. The number $d_{|B|-n}$ satisfies $d_{|B|-n} \leq \max(\log(|B|)+\log(n))$. After computing $\vec{s}_r$ and $\vec{\Delta}_{r+1}$, checking the condition in line 4, as well as computing the parameter d in line 5, can be done in $O(d_{|B|-n})$ time. The two type II elementary foldings in lines 5 and 8 take $O(|B|+n)$ time (Section C.2.5), and the total time for right-folding applications in the loop in lines 6-7 is $O(|B|)$ (Section C.2.5). Thus, the total running time of the procedure is $O(|B|+n)$.

Overall Running Time

Let $\vec{n} = [n_1, n_2, \ldots, n_k]$ be the input vector for the algorithm. Denote $N = \sum\limits_{1 \leq l \leq k} n_l$, and note that N is the length of the output string α in case the algorithm does not return "FAILED". It is simple to assert that besides operations conducted within the FOLD procedure, Algorithm SEARCH-BFB performs $O(N)$ operations. For every $1 \leq l \leq k$, FOLD is called once by the BFB algorithm over the collection B^{l+1} of size n_{l+1} and the integer n_l, and runs in $O(n_{l+1}+n_l)$ time (Section C.2.5). Summing the running time of FOLD for $l = k$ down to 1, its overall running time, as well as the overall running time of Algorithm SEARCH-BFB, is $O(N)$.

C.3 The Decision Variant

In this section, we describe a simplification of the SEARCH-BFB algorithm which solves the decision variant of the BFB count vector problem. Essentially, this algorithm applies similar steps to those of the search algorithm, yet instead of explicitly maintaining collections B, the algorithm only maintains the signature $\vec{s}$ of B. We assume that the algorithm maintains explicitly only the prefix $\vec{s}_r$ of $\vec{s}$ (for $r = r(B)$) as a linked list, where for $i > r$ the algorithm takes the value 0 whenever it needs using the value s_i.

The fact that the signature modifications applied by Procedure SIGNATURE-FOLD yield identical signatures to those of the collections computed by Procedure FOLD can be asserted from Conclusion 1 and Claims 20 and 25. It may also be asserted that the total number of operations in all calls to Procedure ADD-EMPTY (lines 2, 7, and 11 in Procedure SIGNATURE-FOLD), as well as the computation of $\vec{\Delta}_{d_{n_B - n}}$ in line 4, checking the condition in line 5, and computing d in line 6, is $O(r(B) + r(B')) = O(\log n_B + \log n)$. Besides these operations, Procedure SIGNATURE-FOLD applies additional $O(1)$ operations, hence its total running time is $O(\log n_B + \log n)$. Therefore, the overall running time of Algorithm DECISION-BFB is
$O\left(\sum_{0 \le l \le k} (\log n_{l+1} + \log n_l)\right) = O\left(\sum_{0 \le l \le k} \log n_l\right) = O(\tilde{N})$, where $\tilde{N}$ is the number of bits in the representation of the input vector $\vec{n}$. A more involved amortized analysis, omitted from this text, may show that the algorithm performs $O(\tilde{N})$ bit operations, hence being strictly linear with respect to its input length.

C.4 The Distance Variant

This section gives Algorithm DISTANCE-BFB for solving the distance variant of the BFB count vector problem. As a matter of fact, the presented algorithm solves the problem for every suffix $\vec{n}^l = [n_l, n_{l+1}, \ldots, n_k]$ of the input vector $\vec{n} = [n_1, n_2, \ldots, n_k]$.

For a vector $\vec{n} = [n_1, n_2, \ldots, n_k]$ of length k and an integer m, denote by $[m, \vec{n}]$ the $(k+1)$-length vector $[m, n_1, n_2, \ldots, n_k]$. The algorithm is generic and may work with any vector distance measure δ, provided that for any equal-length three vectors $\vec{n}$, $\vec{n}'$, and $\vec{n}''$ such that $\delta(\vec{n}, \vec{n}') \le \delta(\vec{n}, \vec{n}'')$, (1) $\delta(\vec{n}', \vec{n}') = \delta(\vec{n}'', \vec{n}'') = 0 \le \delta(\vec{n}, \vec{n}') \le \delta(\vec{n}', \vec{n}'') \le 1$, and (2) for any pair of numbers m and m', $\delta([m, \vec{n}], [m', \vec{n}']) \le \delta([m, \vec{n}], [m', \vec{n}''])$. For some precision parameter $0 \le \eta < 1$, the algorithm finds the exact solution for the distance variant of the BFB count vector problem for every suffix of the input vector for which the solution is at most η, and returns the approximated solution 1 to suffixes for which the solution is greater than η.

Similar to Algorithms SEARCH-BFB and DESCISION-BFB, Algorithm

DISTANCE-BFB runs k iterations on an input vector $\vec{n} = [n_1, n_2, \ldots, n_k]$, indexed from k down to 1. At the end of iteration l, the algorithm computes a collection S^l containing elements of the form $(\vec{n}^i = [n^i_l, n^i_{l+1}, \ldots, n^i_k], \vec{s}^i)$, where $\vec{s}^i$ is the minimum signature of an l-block collection B^i admitting the count vector $\vec{n}^i$, and $\delta(\vec{n}^l, \vec{n}^i) \leq \eta$. It is guaranteed that for every BFB count vector $\vec{n}^j = [n^j_l, n^j_{l+1}, \ldots, n^j_k]$ such that $\delta(\vec{n}^l, \vec{n}^j) \leq \eta$ and every l-block collection B^j admitting $\vec{n}^j$, S^l contains a pair $(\vec{n}^i, \vec{s}^i)$ such that $\delta(\vec{n}^l, \vec{n}^i) \leq \delta(\vec{n}^l, \vec{n}^j)$ and $\vec{s}^i \leq \vec{s}^j$.

Consider the signature $\vec{s}$ of a collection B of size n. It is simple to show that $r(B) \leq \log n + 1$, and that $-n < s_i \leq n$ for every $0 \leq i \leq r$. Therefore, $\vec{s}$ can be represented by $O(\log^2 n)$ bits, and so the number of different signatures of collections of size n is upper bounded by $2^{O(\log^2 n)}$. In addition, under realistic assumptions, we may assume that the number of different values n examined in line 6 of Algorithm DISTANCE-BFB bounded by $2^{O(\log^2 n_l)}$, since this number should approximate the count n_l (for example, using the Poisson δ function described in Chapter 4, it is possible to show that for every value of n_l and $\vec{n}'^i$ and for $n \geq 20n_l$, $\delta(\vec{n}^l, [n, \vec{n}'^i]) > 1 - 10^{-6}$, thus choosing $\eta = 1 - 10^{-6}$ guarantees that the loop in lines 6-9 is being executed less than $20n_l$ times for every $(\vec{n}'^i, \vec{s}^i) \in S^{l+1}$). Due to the condition in line 7, every possible signature $\vec{s}$ appears at most once in some pair in S^l, thus the size of S^l is bounded by $2^{O(\log^2 n_l)}$. It is straightforward to observe that the total number of operations in the loop in lines 7-9 is also $2^{O(\log^2 n_l)}$, and so the total running time of the algorithm is bounded by $\sum\limits_{1 \leq l \leq k} 2^{O(\log^2 n_l)} \leq 2^{O(\log^2 N)} = N^{O(\log N)}$.

C.5 Chromosome simulation details

Each chromosome pair was modeled as two sequences of 100,000,000 ordered bases. Then fifty rearrangement were introduced to each chromosome independently. Each rearrangement type was chosen randomly from deletion, inversion, and duplication, according to a distribution. Thus, both balanced and unbalanced rearrangements were used to simulate the chromosomes. If the chosen rearrangement was a duplication, then it was decided whether or not the duplication would be tandem and whether or not it would be inverted. Tandem duplications would be inserted adjacent to the original chromosome segment, and inverted duplications would have the new duplicated segment reversed with respect to the original segment.

Two rearrangement type regimes were used. In the first regime, referred to as “evendup” in the supplemental data, each rearrangement was a duplication, inversion, or deletion with probability .5, .25, and .25 respectively. Duplications had a 50% chance of being tandem and, independently, a 50% chance of being inverted. In the second regime, called “highdup” in the supplemental data, the probability of duplication, inversion, and deletion were $\frac{7}{11}$, $\frac{2}{11}$, and $\frac{1}{11}$. The probability of a duplication being tandem or inverted was .9 and .9. This second regime was created because in the first, fold-back inversions occur infrequently. The second regime allowed us to examine tests for BFB when an alternative mechanism is creating many fold-back inversions.

The size of each non-BFB rearrangement was chosen from a normal distribution bounded at zero with mean 10,000 and a variance of 10,000,000. Rearrangements were introduced sequentially in each chromosome. For chromsomes in which BFB was simulated, consecutive rounds of BFB were introduced after one of the fifty non-BFB rearrangments. The number of BFB rounds varied from two to ten. Each BFB round consisted of a prefix of the chromosome undergoing a tandem inverted duplication. The size of the prefix was selected from a normal distribution with a mean of zero and a standard deviation of one tenth of the length of the chromosome.

After each chromosome in the pair was rearranged, the copy numbers and breakpoints were combined as one would expect from experimental evidence.

C.6 Cancer cell line results

We identified count vectors on three chromosomes from the 746 cancer cell lines that were long and nearly consistent with BFB. The observed count vectors along with the nearest count vector consistent with BFB are shown below.

Cell line: AU565 Tissue: bone

Chromosome 8 between 72.5 MB and 80.0 MB

```
Observed 4,8,14,10,8,14,9,13,7,12,9,7
Fit      4,8,14,10,8,14,9,13,7,13,9,7
```

Cell line: PC-3 Tissue: prostate

Chromosome 10 between 60 MB and 82 MB

```
Observed 6,10,14, 9,6,9,13,9,5,9,3,14
Fit      6,10,14,10,6,9,13,9,5,9,3,15
```

Cell line: MG-63 Tissue: bone

Chromosome 8 between 112 MB and 121 MB

```
Observed 10,6,8,14,11,14,9,8,13,9,13,9,7
Fit      10,6,8,14,11,15,9,9,13,9,13,9,7
```

C.7 ROC curves for varying simulation parameters

Below are the ROC curves, similar to those in Figure 4 of Chapter 4, for many different simulation and test parameters.

C.8 Pancreatic cancer data analysis pipeline

Figure C.6 shows a graphical layout of the analysis.

C.9 Possible arrangement of segments on BFB -rearranged chromosome 12

```
**Procedure**: FOLD$(B, n)$

**Input**: An $l$-BFB palindrome collection $B$ and an integer $n \geq 0$.
**Output**: A minimum signature folding $B'$ of $B$ such that $|B'| = n$, or the string "FAILD" if
    there is no such $B'$.

1  If $|B| \leq n$ then return $B + (n - |B|)\varepsilon$.
2  Else
3      Let $\vec{s} = \vec{s}(B)$ and $\vec{\Delta} = \vec{\Delta}(B)$.
4      If there exists $0 \leq d \leq d_{|B|-n}$ such that $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$ then
5          Let $d$ be the maximum integer sustaining the condition above. Set $B' \leftarrow B + 2^d\varepsilon$.
6          While $|B'| > n$ do
7              Set $B' \leftarrow$ RIGHT-FOLD$(B')$.
8          Set $B' \leftarrow B' + (n - |B'|)\varepsilon$.
9          Return $B'$
10     Else return "FAILED"
```

```
**Procedure**: RIGHT-FOLD$(B)$

**Input**: An $l$-BFB palindrome collection $B$.
**Precondition**: Let $\left\langle \vec{B}, \vec{L}, \vec{H} \right\rangle$ be the decomposition of $B$, and $r = r(B)$. There is an integer
    $0 \leq g < r$ such that $H_g \neq \emptyset$, $L_g \neq \emptyset$, and for every $g < i < r$, $H_i = \emptyset$ and
    $L_i \neq \emptyset$.
**Output**: A folding $B'$ of $B$ of size $|B| - 2^r$.

1  Let $g$ be an integer as implied from the precondition (note that $g$ is unique), $\beta$ a minimal
   element in $H_g$, $A = \{\alpha_1, 2\alpha_2, \ldots, 2^{r-g-1}\alpha_{r-g}\}$ a convexed $l$-collection such that
   $\alpha_i \in L_{g+i-1}$ for each $1 \leq i \leq r - g$ and $\alpha_1$ is a minimal element in $L_g$, and $\alpha = \beta\gamma_A\beta$.
2  Return the collection $B' = B - 2^g(2\beta + A) + 2^g\alpha$.
```

```
Algorithm: DECISION-BFB($\vec{n}$)
Input: A count vector $\vec{n} = [n_1, n_2, \dots, n_k]$.
Output: "TRUE" if $\vec{n}$ is a BFB count vector, and "FAILED" if otherwise.
1 Set $n_{k+1} \leftarrow 0$ and $\vec{s}^{k+1} \leftarrow \vec{0}$.
2 For $l \leftarrow k$ down to 1 do
3     Apply SIGNATURE-FOLD($\vec{s}^{l+1}, n_{l+1}, n_l$). If this operation has failed, return
      "FALSE".
4     Otherwise, set $\vec{s}^{l}$ to be the returned value from the call to
      SIGNATURE-FOLD($\vec{s}^{l+1}, n_{l+1}, n_l$).
5 Apply SIGNATURE-FOLD($\vec{s}^{1}, n_1, 1$). If this operation has failed, return "FALSE", and
  otherwise return "TRUE".
```

```
Procedure: SIGNATURE-FOLD($\vec{s}, n_B, n$)
Input: The signature $\vec{s}$ and size $n_B$ of an $l$-block collection $B$ and an integer $n \geq 0$.
Output: The minimum signature $\vec{s}'$ of a folding $B'$ of $B$ such that $|B'| = n$, or the string
        "FAILD" if there is no such $B'$.
1 If $n_B \leq n$ then
2     return ADD-EMPTY($\vec{s}, n_B, n - n_B$).
3 Else
4     Compute the prefix $\vec{\Delta}_{d_{n_B - n}}$ of $\vec{\Delta}(B)$.
5     If there exists $0 \leq d \leq d_{n_B - n}$ such that $n \geq 2^d \max(s_d + 1, 0) + \Delta_d$ then
6         Let $d$ be the maximum integer sustaining the condition above.
7         Set $\vec{s}' \leftarrow$ ADD-EMPTY($\vec{s}, n_B, 2^d$), and $n_{B'} \leftarrow n_B + 2^d$.
8         If $n \geq n_{B'} + 2^{d+1} s'_{d+1}$ then
9             Set $s'_{d+1} \leftarrow s'_{d+1} + \left\lceil \frac{n_{B'} - n}{2^{d+1}} \right\rceil$.
10            Set $n_{B'} \leftarrow \Delta_d + 2^d \mathrm{abs}(s'_d) + 2^{d+1} \mathrm{abs}(s'_{d+1})$.
11            Set $\vec{s}' \leftarrow$ ADD-EMPTY($\vec{s}', n_{B'}, n - n_{B'}$).
12        Else
13            Set $s'_d \leftarrow s'_d + \frac{n_{B'} - n}{2^d} + 2 s'_{d+1}$.
14            Set $s'_{d+1} \leftarrow 0$.
15        return $\vec{s}'$.
16    Else return "FAILED"
```

```
**Procedure**: ADD-EMPTY($(\vec{s}, n_B, m)$)
**Input**: The signature $\vec{s}$ and size $n_B$ of an $l$-BFB palindrome collection $B$ containing no $\varepsilon$
    elements, and an integer $m \geq 0$.
**Output**: The signature $\vec{s}'$ of the folding $B' = B + m\varepsilon$ of $B$.
1 **If** $n_B = m$ **then**
2     **return** $\vec{s}$
3 **Else**
4     Let $d = d_{n-n_B}$, and set the prefix $\vec{s}'_{d-1}$ to be the copy of the prefix $\vec{s}'_{d-1}$ of $\vec{s}$.
5     Set $s'_d \leftarrow s_d + 1$.
6     Compute $\vec{\Delta}'_d = \sum_{0 \leq i < d} 2^i \text{abs}(s_i)$.
7     Set $s'_{d+1} \leftarrow \frac{\vec{\Delta}_d + 2^d \text{abs}(s'_d) - n}{2^{d+1}}$. // All values $s'_i$ for $i > d+1$ are implicitly
      set to 0.
8     **return** $\vec{s}'$.
```

```
Algorithm: DISTANCE-BFB($\vec{n}, \eta$)
Input: A count vector $\vec{n} = [n_1, n_2, \ldots, n_k]$, and a precision parameter $0 \leq \eta < 1$.
Output: For every $1 \leq l \leq k$, the algorithm reports the minimum distance $\delta_l$ of the suffix
        $\vec{n}^l = [n_l, n_{l+1}, \ldots, n_k]$ of $\vec{n}$ from a BFB count vector, in case this distance is at
        most $\eta$.
// Collections of the form $S^l$ contain pairs $(\vec{n}'^i = [n'_l, n'_{l+1}, \ldots, n'_k], \vec{s}^i)$,
   where $\vec{s}^i$ is the minimum signature of an $l$-block collection $B^i$
   admitting the count vector $\vec{n}'^i$, and $\delta(\vec{n}^l, \vec{n}'^i) \leq \eta$.
1  Set $S^{k+1}$ be a collection containing only the pair $(\vec{0}, 0)$.
2  For $l \leftarrow k$ down to 1 do
3      Set $\delta_l \leftarrow 1$.
4      Set $S^l \leftarrow \emptyset$.
5      For each $(\vec{n}'^i = [n'_{l+1}, \ldots, n'_k], \vec{s}^i) \in S^{l+1}$ do
6          For each $n \geq 1$ such that $\delta(\vec{n}^l, [n, \vec{n}'^i]) \leq \eta$ do
7              If SIGNATURE-FOLD$(\vec{n}'^i, n'_{l+1}, n) = \vec{s}$, IS-PALINDROMIC$(\vec{s})$, and for all
               $(\vec{n}'^j, \vec{s}^j) \in S^l$ such that $\delta(\vec{n}^l, \vec{n}'^j) \leq \delta(\vec{n}^l, [n, \vec{n}'^i])$ it is true that $\vec{s} < \vec{n}'^j$ then
8                  Add $([n, \vec{n}'^i], \vec{s})$ to $S^l$.
9                  Set $\delta_l \leftarrow \min(\delta_l, \delta(\vec{n}^l, [n, \vec{n}'^i]))$.
10     Report $\delta_l$.
```

```
Procedure: IS-PALINDROMIC($\vec{s}$)
Input: The signature $\vec{s}$ of an $l$-BFB palindrome collection $B$.
Output: "TRUE" if it is possible to concatenate all elements in $B$ into a single $l$-BFB
        palindrome, and "FALSE" otherwise.
1  Compute the prefix $\vec{\Delta}_{r+1}$ of $\vec{\Delta}(B)$ for $r = r(B)$. // Note that $|B| = \Delta_{r+1}$
2  If there exists $0 \leq d \leq d_{\Delta_{r+1}-1}$ such that $1 \geq 2^d \max(s_d + 1, 0) + \Delta_d$ then return "TRUE".
3  Else return "False".
```

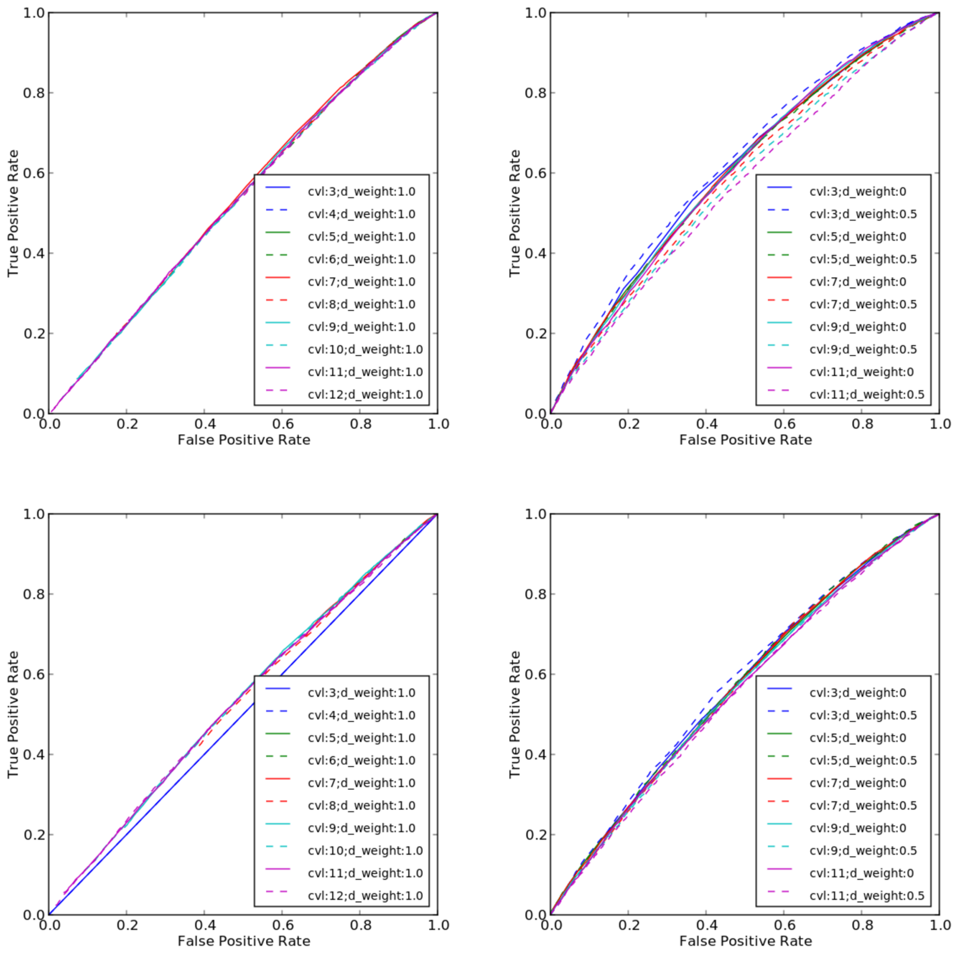


Figure C.1: ROC curves for simulations with 2 rounds of BFB. Clockwise from the upper left, evendup background with no use of fold-back fraction, evendup background using fold-backs, highdup background using fold-backs, highdup background with no use of fold-back fraction.

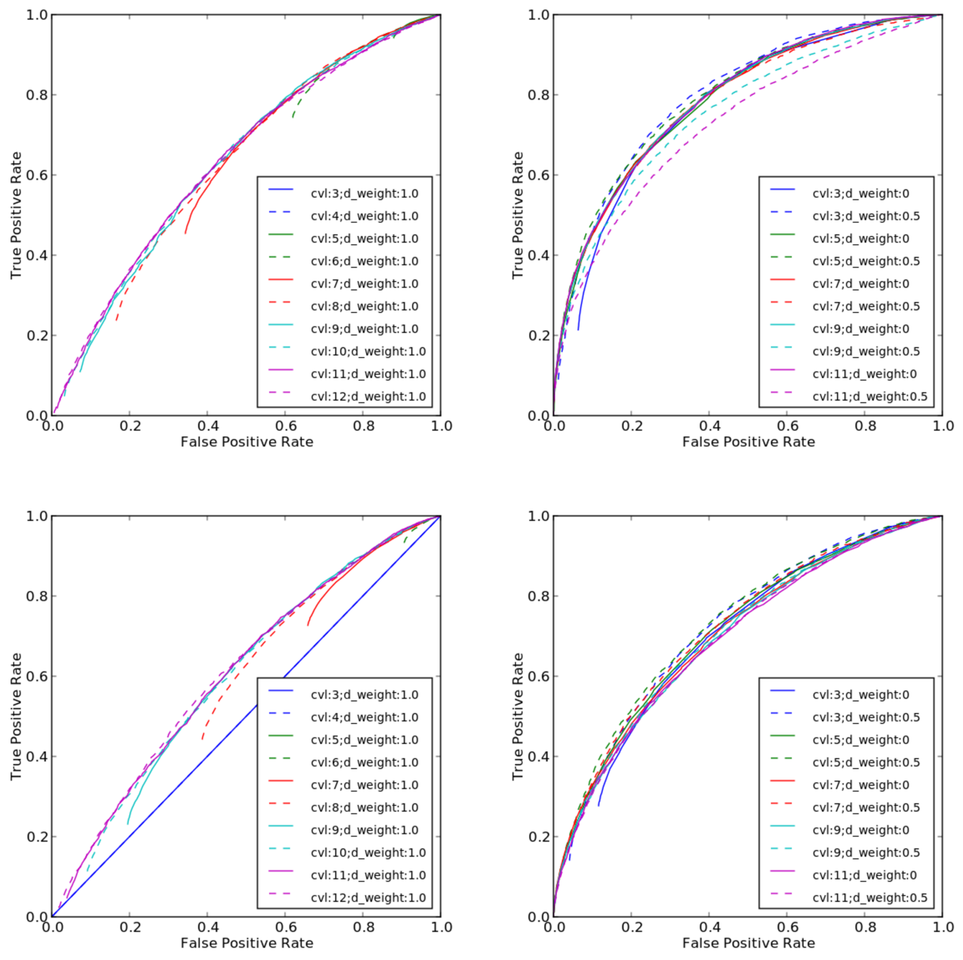


Figure C.2: ROC curves for simulations with 4 rounds of BFB. Clockwise from the upper left, evendup background with no use of fold-back fraction, evendup background using fold-backs, highdup background using fold-backs, highdup background with no use of fold-back fraction.

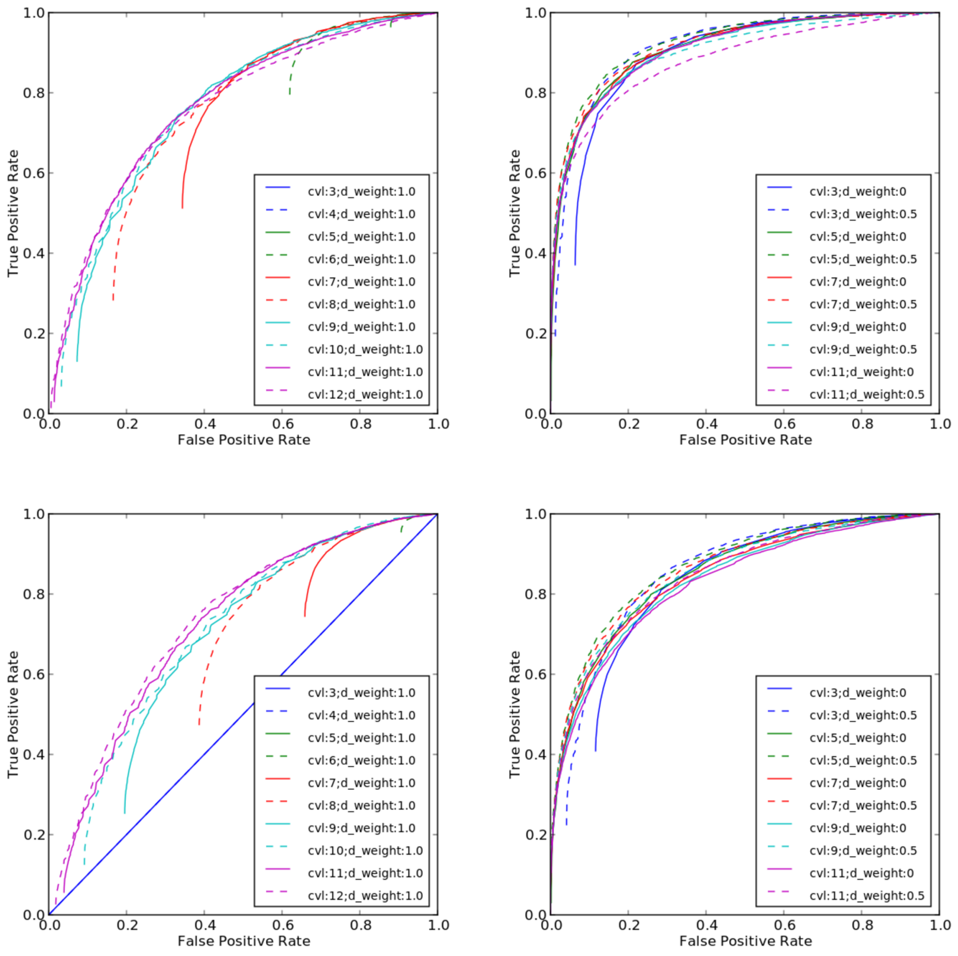


Figure C.3: ROC curves for simulations with 6 rounds of BFB. Clockwise from the upper left, evendup background with no use of fold-back fraction, evendup background using fold-backs, highdup background using fold-backs, highdup background with no use of fold-back fraction.

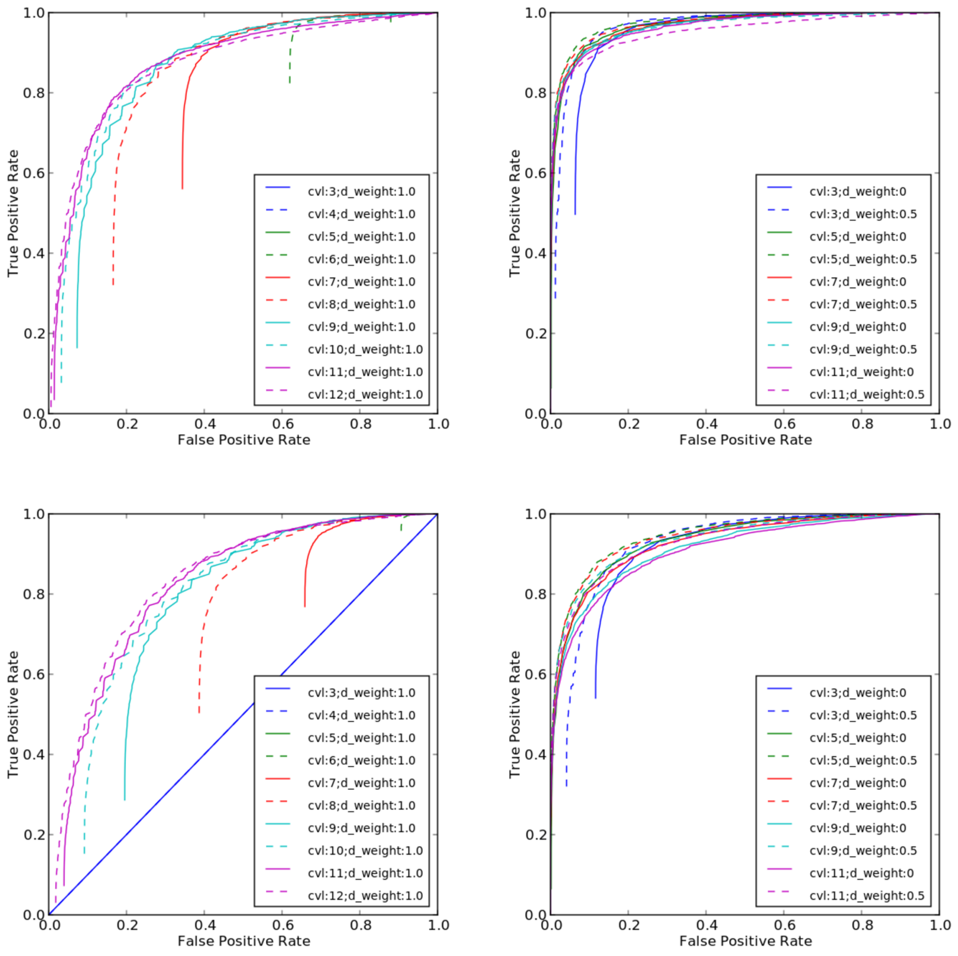


Figure C.4: ROC curves for simulations with 8 rounds of BFB. Clockwise from the upper left, evendup background with no use of fold-back fraction, evendup background using fold-backs, highdup background using fold-backs, highdup background with no use of fold-back fraction.

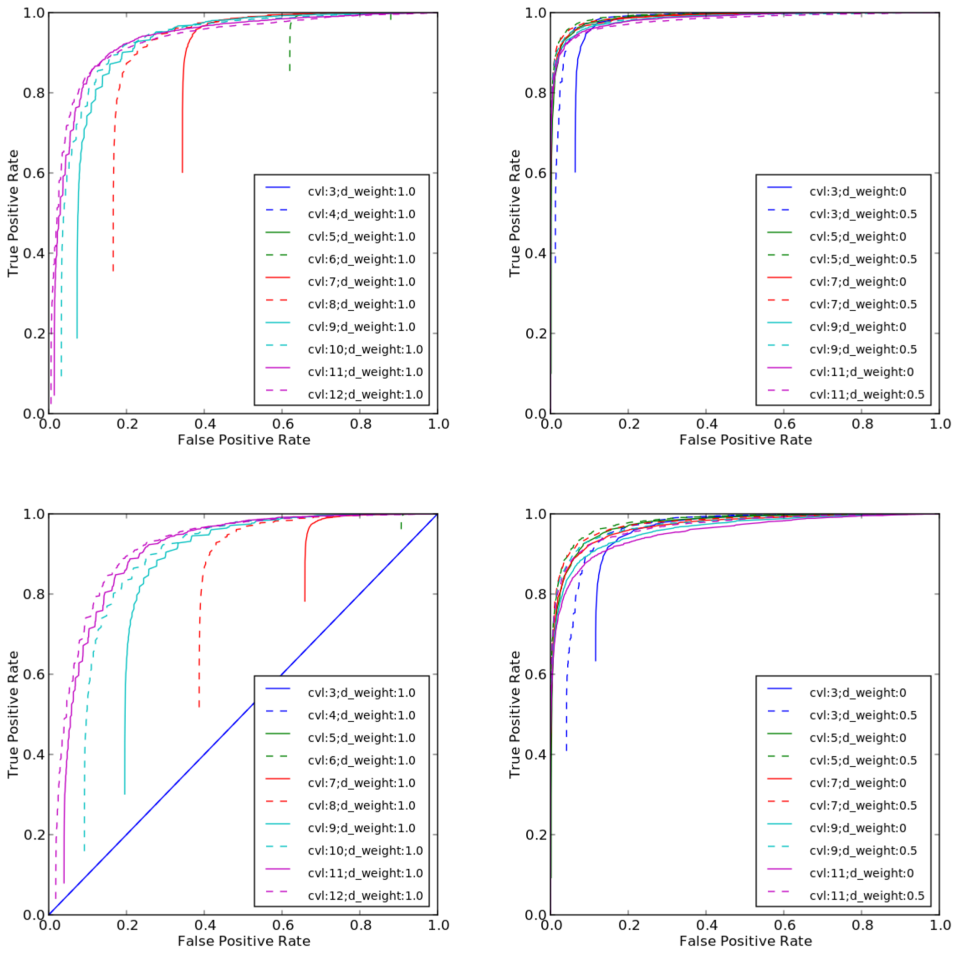


Figure C.5: ROC curves for simulations with 10 rounds of BFB. Clockwise from the upper left, evendup background with no use of fold-back fraction, evendup background using fold-backs, highdup background using fold-backs, highdup background with no use of fold-back fraction.

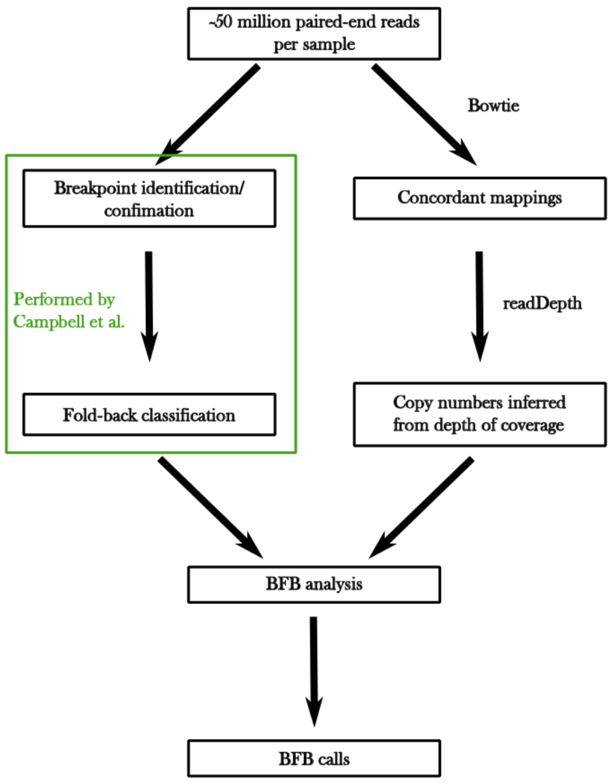

~50 million paired-end reads per sample
Bowtie
Breakpoint identification/ confimation
Concordant mappings
Performed by Campbell et al.
readDepth
Fold-back classification
Copy numbers inferred from depth of coverage
BFB analysis
BFB calls

Figure C.6: Graphical representation of the analysis performed with the pancreatic cancer paired-end sequencing data.

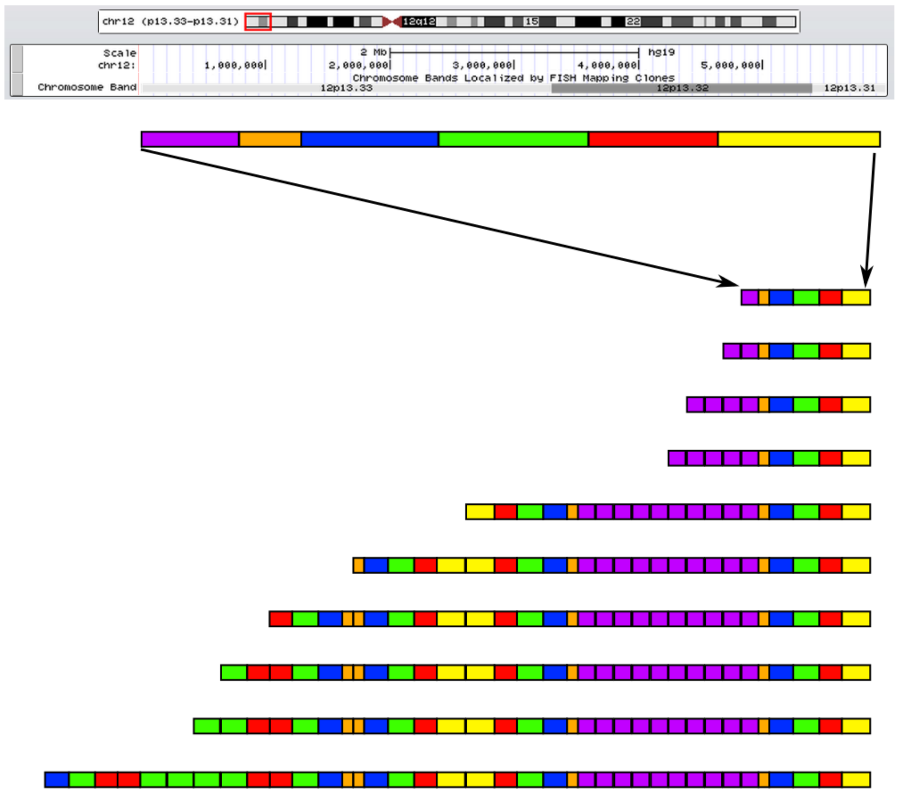


Figure C.7: Plausible BFB cycles that could lead to the copy counts observed in chromosome 12 of pancreatic cancer sample PD3641.

Bibliography

[1] P. Akiva, A. Toporik, S. Edelheit, Y. Peretz, A. Diber, R. Shemesh, A. Novik, and R. Sorek. Transcription-mediated gene fusion in the human genome. *Genome Res.*, 16:30–36, Jan 2006.

[2] A. Ameur, A. Wetterbom, L. Feuk, and U. Gyllensten. Global and unbiased detection of splice junctions from RNA-seq data. *Genome Biol.*, 11:R34, 2010.

[3] S. E. Artandi and R. A. DePinho. Telomeres and telomerase in cancer. *Carcinogenesis*, 31:9–18, Jan 2010.

[4] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Res.*, 36:25–30, Jan 2008.

[5] M. F. Berger, J. Z. Levin, K. Vijayendran, A. Sivachenko, X. Adiconis, J. Maguire, L. A. Johnson, J. Robinson, R. G. Verhaak, C. Sougnez, R. C. Onofrio, L. Ziaugra, K. Cibulskis, E. Laine, J. Barretina, W. Winckler, D. E. Fisher, G. Getz, M. Meyerson, D. B. Jaffe, S. B. Gabriel, E. S. Lander, R. Dummer, A. Gnirke, C. Nusbaum, and L. A. Garraway. Integrative analysis of the melanoma transcriptome. *Genome Res.*, 20:413–427, Apr 2010.

[6] G. R. Bignell, C. D. Greenman, H. Davies, A. P. Butler, S. Edkins, J. M. Andrews, G. Buck, L. Chen, D. Beare, C. Latimer, S. Widaa, J. Hinton, C. Fahey, B. Fu, S. Swamy, G. L. Dalgliesh, B. T. Teh, P. Deloukas, F. Yang, P. J. Campbell, P. A. Futreal, and M. R. Stratton. Signatures of mutation and selection in the cancer genome. *Nature*, 463(7283):893–898, Feb 2010.

[7] G. R. Bignell, T. Santarius, J. C. Pole, A. P. Butler, J. Perry, E. Pleasance, C. Greenman, A. Menzies, S. Taylor, S. Edkins, P. Campbell, M. Quail, B. Plumb, L. Matthews, K. McLay, P. A. Edwards, J. Rogers, R. Wooster, P. A. Futreal, and M. R. Stratton. Architectures of somatic genomic rearrangement in human cancer amplicons at sequence-level resolution. *Genome Res.*, 17:1296–1303, Sep 2007.

[8] J. H. Bullard, E. Purdom, K. D. Hansen, and S. Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 11:94, 2010.

[9] P. J. Campbell, S. Yachida, L. J. Mudie, P. J. Stephens, E. D. Pleasance, L. A. Stebbings, L. A. Morsberger, C. Latimer, S. McLaren, M. L. Lin, D. J. McBride, I. Varela, S. A. Nik-Zainal, C. Leroy, M. Jia, A. Menzies, A. P. Butler, J. W. Teague, C. A. Griffin, J. Burton, H. Swerdlow, M. A. Quail, M. R. Stratton, C. Iacobuzio-Donahue, and P. A. Futreal. The patterns and dynamics of genomic instability in metastatic pancreatic cancer. *Nature*, 467(7319):1109–1113, Oct 2010.

[10] P. Carninci. Is sequencing enlightenment ending the dark age of the transcriptome? *Nat. Methods*, 6:711–713, Oct 2009.

[11] P. Carninci, T. Kasukawa, S. Katayama, J. Gough, M. C. Frith, N. Maeda, R. Oyama, T. Ravasi, B. Lenhard, C. Wells, R. Kodzius, K. Shimokawa, et al. The transcriptional landscape of the mammalian genome. *Science*, 309:1559–1563, Sep 2005.

[12] A. M. Carr, A. L. Paek, and T. Weinert. DNA replication: failures and inverted fusions. *Semin. Cell Dev. Biol.*, 22(8):866–874, Oct 2011.

[13] N. P. Carter. Methods and strategies for analyzing copy number variation using DNA microarrays. *Nat. Genet.*, 39(7 Suppl):16–21, Jul 2007.

[14] C. Chiang, J. C. Jacobsen, C. Ernst, C. Hanscom, A. Heilbut, I. Blumenthal, R. E. Mills, A. Kirby, A. M. Lindgren, S. R. Rudiger, C. J. McLaughlan, C. S. Bawden, S. J. Reid, R. L. Faull, R. G. Snell, I. M. Hall, Y. Shen, T. K. Ohsumi, M. L. Borowsky, M. J. Daly, C. Lee, C. C. Morton, M. E. MacDonald, J. F. Gusella, and M. E. Talkowski. Complex reorganization and predominant non-homologous repair following chromosomal breakage in karyotypically balanced germline rearrangements and transgenic integration. *Nat. Genet.*, 44(4):390–397, Apr 2012.

[15] D. Y. Chiang, G. Getz, D. B. Jaffe, M. J. O'Kelly, X. Zhao, S. L. Carter, C. Russ, C. Nusbaum, M. Meyerson, and E. S. Lander. High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nat. Methods*, 6(1):99–103, Jan 2009.

[16] R. de Cid, E. Riveira-Munoz, P. L. Zeeuwen, J. Robarge, W. Liao, E. N. Dannhauser, E. Giardina, P. E. Stuart, R. Nair, C. Helms, G. Escaramis, E. Ballana, G. Martin-Ezquerra, M. den Heijer, M. Kamsteeg, I. Joosten, E. E. Eichler, C. Lazaro, R. M. Pujol, L. Armengol, G. Abecasis, J. T. Elder, G. Novelli, J. A. Armour, P. Y. Kwok, A. Bowcock, J. Schalkwijk, and X. Estivill. Deletion of the late cornified envelope LCE3B and LCE3C genes as a susceptibility factor for psoriasis. *Nat. Genet.*, 41(2):211–215, Feb 2009.

[17] R. A. DePinho and K. Polyak. Cancer chromosomes in crisis. *Nat. Genet.*, 36(9):932–934, Sep 2004.

[18] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 2009.

[19] B. J. Druker, M. Talpaz, D. J. Resta, B. Peng, E. Buchdunger, J. M. Ford, N. B. Lydon, H. Kantarjian, R. Capdeville, S. Ohno-Jones, and C. L. Sawyers. Efficacy and safety of a specific inhibitor of the BCR-ABL tyrosine kinase in chronic myeloid leukemia. *N. Engl. J. Med.*, 344:1031–1037, Apr 2001.

[20] P. A. Edwards. Fusion genes and chromosome translocations in the common epithelial cancers. *J. Pathol.*, 220:244–254, Jan 2010.

[21] G. J. Faulkner, A. R. Forrest, A. M. Chalk, K. Schroder, Y. Hayashizaki, P. Carninci, D. A. Hume, and S. M. Grimmond. A rescue strategy for multimapping short sequence tags refines surveys of transcriptional activity by CAGE. *Genomics*, 91:281–288, Mar 2008.

[22] K. A. Frazer, D. G. Ballinger, D. R. Cox, D. A. Hinds, L. L. Stuve, R. A. Gibbs, J. W. Belmont, A. Boudreau, P. Hardenbol, S. M. Leal, S. Pasternak, D. A. Wheeler, T. D. Willis, F. Yu, H. Yang, C. Zeng, Y. Gao, H. Hu, W. Hu, C. Li, W. Lin, S. Liu, H. Pan, X. Tang, J. Wang, W. Wang, J. Yu, B. Zhang, Q. Zhang, H. Zhao, H. Zhao, J. Zhou, S. B. Gabriel, R. Barry, B. Blumenstiel, A. Camargo, M. Defelice, M. Faggart, M. Goyette, S. Gupta, J. Moore, H. Nguyen, R. C. Onofrio, M. Parkin, J. Roy, E. Stahl, E. Winchester, L. Ziaugra, D. Altshuler, Y. Shen, Z. Yao, W. Huang, X. Chu, Y. He, L. Jin, Y. Liu, Y. Shen, W. Sun, H. Wang, Y. Wang, Y. Wang, X. Xiong, L. Xu, M. M. Waye, S. K. Tsui, H. Xue, J. T. Wong, L. M. Galver, J. B. Fan, K. Gunderson, S. S. Murray, A. R. Oliphant, M. S. Chee, A. Montpetit, F. Chagnon, V. Ferretti, M. Leboeuf, J. F. Olivier, M. S. Phillips, S. Roumy, C. Sallee, A. Verner, T. J. Hudson, P. Y. Kwok, D. Cai, D. C. Koboldt, R. D. Miller, L. Pawlikowska, P. Taillon-Miller, M. Xiao, L. C. Tsui, W. Mak, Y. Q. Song, P. K. Tam, Y. Nakamura, T. Kawaguchi, T. Kitamoto, T. Morizono, A. Nagashima, Y. Ohnishi, A. Sekine, T. Tanaka, T. Tsunoda, P. Deloukas, C. P. Bird, M. Delgado, E. T. Dermitzakis, R. Gwilliam, S. Hunt, J. Morrison, D. Powell, B. E. Stranger, P. Whittaker, D. R. Bentley, M. J. Daly, P. I. de Bakker, J. Barrett, Y. R. Chretien, J. Maller, S. McCarroll, N. Patterson, I. Pe’er, A. Price, S. Purcell, D. J. Richter, P. Sabeti, R. Saxena, S. F. Schaffner, P. C. Sham, P. Varilly, D. Altshuler, L. D. Stein, L. Krishnan, A. V. Smith, M. K. Tello-Ruiz, G. A. Thorisson, A. Chakravarti, P. E. Chen, D. J. Cutler, C. S. Kashuk, S. Lin, G. R. Abecasis, W. Guan, Y. Li, H. M. Munro, Z. S. Qin, D. J. Thomas, G. McVean, A. Auton, L. Bottolo, N. Cardin, S. Eyheramendy, C. Freeman, J. Marchini, S. Myers, C. Spencer, M. Stephens, P. Donnelly, L. R. Cardon, G. Clarke, D. M. Evans, A. P. Morris, B. S. Weir, T. Tsunoda, J. C. Mullikin, S. T. Sherry, M. Feolo, A. Skol, H. Zhang, C. Zeng, H. Zhao, I. Matsuda, Y. Fukushima, D. R. Macer, E. Suda, C. N. Rotimi, C. A. Adebamowo, I. Ajayi, T. Aniagwu, P. A. Marshall, C. Nkwodimmah, C. D. Royal, M. F. Leppert, M. Dixon, A. Peiffer, R. Qiu, A. Kent, K. Kato, N. Niikawa, I. F. Adewole, B. M. Knoppers, M. W. Foster, E. W. Clayton, J. Watkin, R. A. Gibbs, J. W. Belmont, D. Muzny, L. Nazareth, E. Sodergren, G. M. Weinstock, D. A. Wheeler, I. Yakub, S. B. Gabriel, R. C. Onofrio,

D. J. Richter, L. Ziaugra, B. W. Birren, M. J. Daly, D. Altshuler, R. K. Wilson, L. L. Fulton, J. Rogers, J. Burton, N. P. Carter, C. M. Clee, M. Griffiths, M. C. Jones, K. McLay, R. W. Plumb, M. T. Ross, S. K. Sims, D. L. Willey, Z. Chen, H. Han, L. Kang, M. Godbout, J. C. Wallenburg, P. L'Archeveque, G. Bellemare, K. Saeki, H. Wang, D. An, H. Fu, Q. Li, Z. Wang, R. Wang, A. L. Holden, L. D. Brooks, J. E. McEwen, M. S. Guyer, V. O. Wang, J. L. Peterson, M. Shi, J. Spiegel, L. M. Sung, L. F. Zacharia, F. S. Collins, K. Kennedy, R. Jamieson, and J. Stewart. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449(7164):851–861, Oct 2007.

[23] T. R. Gingeras. Implications of chimaeric non-co-linear transcripts. *Nature*, 461:206–211, Sep 2009.

[24] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, May 1997.

[25] Y. Hahn, T. K. Bera, K. Gehlhaus, I. R. Kirsch, I. H. Pastan, and B. Lee. Finding fusion genes resulting from chromosome rearrangement by analyzing the expressed sequence databases. *Proc. Natl. Acad. Sci. U.S.A.*, 101:13257–13261, Sep 2004.

[26] D. Hanahan and R. A. Weinberg. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–674, Mar 2011.

[27] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13, May 1984.

[28] P. J. Hastings, J. R. Lupski, S. M. Rosenberg, and G. Ira. Mechanisms of change in gene copy number. *Nat. Rev. Genet.*, 10(8):551–564, Aug 2009.

[29] J. Hicks, A. Krasnitz, B. Lakshmi, N. E. Navin, M. Riggs, E. Leibu, D. Esposito, J. Alexander, J. Troge, V. Grubor, S. Yoon, M. Wigler, K. Ye, A. L. Borresen-Dale, B. Naume, E. Schlicting, L. Norton, T. Hagerstrom, L. Skoog, G. Auer, S. Maner, P. Lundin, and A. Zetterberg. Novel patterns of genome rearrangement and their association with survival in breast cancer. *Genome Res.*, 16(12):1465–1479, Dec 2006.

[30] A. M. Hillmer, F. Yao, K. Inaki, W. H. Lee, P. N. Ariyaratne, A. S. Teo, X. Y. Woo, Z. Zhang, H. Zhao, L. Ukil, J. P. Chen, F. Zhu, J. B. So, M. Salto-Tellez, W. T. Poh, K. F. Zawack, N. Nagarajan, S. Gao, G. Li, V. Kumar, H. P. Lim, Y. Y. Sia, C. S. Chan, S. T. Leong, S. C. Neo, P. S. Choi, H. Thoreau, P. B. Tan, A. Shahab, X. Ruan, J. Bergh, P. Hall, V. Cacheux-Rataboul, C. L. Wei, K. G. Yeoh, W. K. Sung, G. Bourque, E. T. Liu, and Y. Ruan. Comprehensive long-span paired-end-tag mapping reveals characteristic patterns of structural variations in epithelial cancer genomes. *Genome Res.*, 21(5):665–675, May 2011.

[31] T. Horiuchi and T. Aigaki. Alternative trans-splicing: a novel mode of pre-mRNA processing. *Biol. Cell*, 98:135–140, Feb 2006.

[32] Y. Hu, K. Wang, X. He, D. Y. Chiang, J. F. Prins, and J. Liu. A Probabilistic Framework for Aligning Paired-end RNA-seq Data. *Bioinformatics*, Jun 2010.

[33] A. J. Iafrate, L. Feuk, M. N. Rivera, M. L. Listewnik, P. K. Donahoe, Y. Qi, S. W. Scherer, and C. Lee. Detection of large-scale variation in the human genome. *Nat. Genet.*, 36(9):949–951, Sep 2004.

[34] Wellcome Trust Sanger Institute. SNP Array Based LOH and Copy Number Analysis SNU-C1 (Chromosome 15). http://www.sanger.ac.uk/cgi-bin/genetics/CGP/cghviewer/CghViewer.cgi?action=DisplayChromosome &chr=15&id=6800, 2012. [Online; accessed 6-March-2013].

[35] J. M. Kidd, G. M. Cooper, W. F. Donahue, H. S. Hayden, N. Sampas, T. Graves, N. Hansen, B. Teague, C. Alkan, F. Antonacci, E. Haugen, T. Zerr, N. A. Yamada, P. Tsang, T. L. Newman, E. Tuzun, Z. Cheng, H. M. Ebling, N. Tusneem, R. David, W. Gillett, K. A. Phelps, M. Weaver, D. Saranga, A. Brand, W. Tao, E. Gustafson, K. McKernan, L. Chen, M. Malig, J. D. Smith, J. M. Korn, S. A. McCarroll, D. A. Altshuler, D. A. Peiffer, M. Dorschner, J. Stamatoyannopoulos, D. Schwartz, D. A. Nickerson, J. C. Mullikin, R. K. Wilson, L. Bruhn, M. V. Olson, R. Kaul, D. R. Smith, and E. E. Eichler. Mapping and sequencing of structural variation from eight human genomes. *Nature*, 453(7191):56–64, May 2008.

[36] M. Kinsella and V. Bafna. Combinatorics of the breakage-fusion-bridge mechanism. *J. Comput. Biol.*, 19(6):662–678, Jun 2012.

[37] K. Kitada and T. Yamasaki. The complicated copy number alterations in chromosome 7 of a lung cancer cell line is explained by a model based on repeated breakage-fusion-bridge cycles. *Cancer Genet. Cytogenet.*, 185:11–19, Aug 2008.

[38] W. P. Kloosterman, V. Guryev, M. van Roosmalen, K. J. Duran, E. de Bruijn, S. C. Bakker, T. Letteboer, B. van Nesselrooij, R. Hochstenbach, M. Poot, and E. Cuppen. Chromothripsis as a mechanism driving complex de novo structural rearrangements in the germline. *Hum. Mol. Genet.*, 20(10):1916–1924, May 2011.

[39] W. P. Kloosterman, M. Hoogstraat, O. Paling, M. Tavakoli-Yaraki, I. Renkens, J. S. Vermaat, M. J. van Roosmalen, S. van Lieshout, I. J. Nijman, W. Roessingh, R. van 't Slot, J. van de Belt, V. Guryev, M. Koudijs, E. Voest, and E. Cuppen. Chromothripsis is a common mechanism driving genomic rearrangements in primary and metastatic colorectal cancer. *Genome Biol.*, 12(10):R103, 2011.

[40] M. Krause and D. Hirsh. A trans-spliced leader sequence on actin mRNA in C. elegans. *Cell*, 49:753–761, Jun 1987.

[41] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, 10:R25, 2009.

[42] J. LEJEUNE, R. TURPIN, and M. GAUTIER. [Chromosomic diagnosis of mongolism]. *Arch. Fr. Pediatr.*, 16:962–963, 1959.

[43] B. Li, V. Ruotti, R. M. Stewart, J. A. Thomson, and C. N. Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26:493–500, Feb 2010.

[44] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, 18:1851–1858, Nov 2008.

[45] X. Li, L. Zhao, H. Jiang, and W. Wang. Short homologous sequences are strongly associated with the generation of chimeric RNAs in eukaryotes. *J. Mol. Evol.*, 68:56–65, Jan 2009.

[46] G. Lim, J. Karaskova, B. Beheshti, B. Vukovic, J. Bayani, S. Selvarajah, S. K. Watson, W. L. Lam, M. Zielenska, and J. A. Squire. An integrated mBAND and submegabase resolution tiling set (SMRT) CGH array analysis of focal amplification, microdeletions, and ladder structures consistent with breakage-fusion-bridge cycle events in osteosarcoma. *Genes Chromosomes Cancer*, 42(4):392–403, Apr 2005.

[47] F. Magrangeas, H. Avet-Loiseau, N. C. Munshi, and S. Minvielle. Chromothripsis identifies a rare and aggressive entity among newly diagnosed multiple myeloma patients. *Blood*, 118(3):675–678, Jul 2011.

[48] C. A. Maher, C. Kumar-Sinha, X. Cao, S. Kalyana-Sundaram, B. Han, X. Jing, L. Sam, T. Barrette, N. Palanisamy, and A. M. Chinnaiyan. Transcriptome sequencing to detect gene fusions in cancer. *Nature*, 458:97–101, Mar 2009.

[49] C. A. Maher, N. Palanisamy, J. C. Brenner, X. Cao, S. Kalyana-Sundaram, S. Luo, I. Khrebtukova, T. R. Barrette, C. Grasso, J. Yu, R. J. Lonigro, G. Schroth, C. Kumar-Sinha, and A. M. Chinnaiyan. Chimeric transcript discovery by paired-end transcriptome sequencing. *Proc. Natl. Acad. Sci. U.S.A.*, 106:12353–12358, Jul 2009.

[50] G. Manacher. A new linear time on-line algorithm for finding the smallest initial palindrome of a string. *J. Assoc. Comput. Mach.*, 22:346–351, July 1975.

[51] S. A. McCarroll, A. Huett, P. Kuballa, S. D. Chilewski, A. Landry, P. Goyette, M. C. Zody, J. L. Hall, S. R. Brant, J. H. Cho, R. H. Duerr, M. S. Silverberg, K. D. Taylor, J. D. Rioux, D. Altshuler, M. J. Daly, and R. J. Xavier. Deletion polymorphism upstream of IRGM associated with altered IRGM expression and Crohn's disease. *Nat. Genet.*, 40(9):1107–1112, Sep 2008.

[52] B. McClintock. The Production of Homozygous Deficient Tissues with Mutant Characteristics by Means of the Aberrant Mitotic Behavior of Ring-Shaped Chromosomes. *Genetics*, 23:315–376, Jul 1938.

[53] B. McClintock. The Stability of Broken Ends of Chromosomes in Zea Mays. *Genetics*, 26:234–282, Mar 1941.

[54] F. Mitelman, B. Johansson, and F. Mertens. Fusion genes and rearranged genes as a linear function of chromosome aberrations in cancer. *Nat. Genet.*, 36:331–334, Apr 2004.

[55] J. J. Molenaar, J. Koster, D. A. Zwijnenburg, P. van Sluis, L. J. Valentijn, I. van der Ploeg, M. Hamdi, J. van Nes, B. A. Westerman, J. van Arkel, M. E. Ebus, F. Haneveld, A. Lakeman, L. Schild, P. Molenaar, P. Stroeken, M. M. van Noesel, I. Ora, E. E. Santo, H. N. Caron, E. M. Westerhout, and R. Versteeg. Sequencing of neuroblastoma identifies chromothripsis and defects in neuritogenesis genes. *Nature*, 483(7391):589–593, Mar 2012.

[56] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat. Methods*, 5:621–628, Jul 2008.

[57] P. A. Northcott, D. J. Shih, J. Peacock, L. Garzia, A. S. Morrissy, T. Zichner, A. M. Stutz, A. Korshunov, J. Reimand, S. E. Schumacher, R. Beroukhim, D. W. Ellison, C. R. Marshall, A. C. Lionel, S. Mack, A. Dubuc, Y. Yao, V. Ramaswamy, B. Luu, A. Rolider, F. M. Cavalli, X. Wang, M. Remke, X. Wu, R. Y. Chiu, A. Chu, E. Chuah, R. D. Corbett, G. R. Hoad, S. D. Jackman, Y. Li, A. Lo, K. L. Mungall, K. M. Nip, J. Q. Qian, A. G. Raymond, N. T. Thiessen, R. J. Varhol, I. Birol, R. A. Moore, A. J. Mungall, R. Holt, D. Kawauchi, M. F. Roussel, M. Kool, D. T. Jones, H. Witt, A. Fernandez-L, A. M. Kenney, R. J. Wechsler-Reya, P. Dirks, T. Aviv, W. A. Grajkowska, M. Perek-Polnik, C. C. Haberler, O. Delattre, S. S. Reynaud, F. F. Doz, S. S. Pernet-Fattet, B. K. Cho, S. K. Kim, K. C. Wang, W. Scheurlen, C. G. Eberhart, M. Fevre-Montange, A. Jouvet, I. F. Pollack, X. Fan, K. M. Muraszko, G. Y. Gillespie, C. Di Rocco, L. Massimi, E. M. Michiels, N. K. Kloosterhof, P. J. French, J. M. Kros, J. M. Olson, R. G. Ellenbogen, K. Zitterbart, L. Kren, R. C. Thompson, M. K. Cooper, B. Lach, R. E. McLendon, D. D. Bigner, A. Fontebasso, S. Albrecht, N. Jabado, J. C. Lindsey, S. Bailey, N. Gupta, W. A. Weiss, L. Bognar, A. Klekner, T. E. Van Meter, T. Kumabe, T. Tominaga, S. K. Elbabaa, J. R. Leonard, J. B. Rubin, L. M. Liau, E. G. Van Meir, M. Fouladi, H. Nakamura, G. Cinalli, M. Garami, P. Hauser, A. G. Saad, A. Iolascon, S. Jung, C. G. Carlotti, R. Vibhakar, Y. S. Ra, S. Robinson, M. Zollo, C. C. Faria, J. A. Chan, M. L. Levy, P. H. Sorensen, M. Meyerson, S. L. Pomeroy, Y. J. Cho, G. D. Bader, U. Tabori, C. E. Hawkins, E. Bouffet, S. W. Scherer, J. T. Rutka, D. Malkin,

S. C. Clifford, S. J. Jones, J. O. Korbel, S. M. Pfister, M. A. Marra, and M. D. Taylor. Subgroup-specific structural variation across 1,000 medulloblastoma genomes. *Nature*, 488(7409):49–56, Aug 2012.

[58] Ankita Patel, Patricia Hixson, Weimin Bi, Caroline Borgan, Marcus Coyle, Danielle Freppon, David Vo, Jacqueline T. O'Hare, Patricia Luke, Chung-Che Chang, and Sau Cheung. Is It Time for Arraycgh to Be the First Line Test for Detection of Chromosome Abnormalities in Hematological Disorders-Example Multiple Myeloma. *BLOOD*, 118(21):1091, NOV 18 2011. 53rd Annual Meeting and Exposition of the American-Society-of-Hematology (ASH)/Symposium on the Basic Science of Hemostasis and Thrombosis, San Diego, CA, DEC 10-13, 2011.

[59] S. Perner, P. L. Wagner, F. Demichelis, R. Mehra, C. J. Lafargue, B. J. Moss, S. Arbogast, A. Soltermann, W. Weder, T. J. Giordano, D. G. Beer, D. S. Rickman, A. M. Chinnaiyan, H. Moch, and M. A. Rubin. EML4-ALK fusion lung cancer: a rare acquired event. *Neoplasia*, 10:298–302, Mar 2008.

[60] K. D. Pruitt, T. Tatusova, and D. R. Maglott. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, 35:D61–65, Jan 2007.

[61] P. Rajan, D. J. Elliott, C. N. Robson, and H. Y. Leung. Alternative splicing and biological heterogeneity in prostate cancer. *Nat Rev Urol*, 6:454–460, Aug 2009.

[62] A. Rajkovic, R. E. Davis, J. N. Simonsen, and F. M. Rottman. A spliced leader is present on a subset of mRNAs from the human parasite Schistosoma mansoni. *Proc. Natl. Acad. Sci. U.S.A.*, 87:8879–8883, Nov 1990.

[63] T. Rausch, D. T. Jones, M. Zapatka, A. M. Stutz, T. Zichner, J. Weischenfeldt, N. Jager, M. Remke, D. Shih, P. A. Northcott, E. Pfaff, J. Tica, Q. Wang, L. Massimi, H. Witt, S. Bender, S. Pleier, H. Cin, C. Hawkins, C. Beck, A. von Deimling, V. Hans, B. Brors, R. Eils, W. Scheurlen, J. Blake, V. Benes, A. E. Kulozik, O. Witt, D. Martin, C. Zhang, R. Porat, D. M. Merino, J. Wasserman, N. Jabado, A. Fontebasso, L. Bullinger, F. G. Rucker, K. Dohner, H. Dohner, J. Koster, J. J. Molenaar, R. Versteeg, M. Kool, U. Tabori, D. Malkin, A. Korshunov, M. D. Taylor, P. Lichter, S. M. Pfister, and J. O. Korbel. Genome sequencing of pediatric medulloblastoma links catastrophic DNA rearrangements with TP53 mutations. *Cell*, 148(1-2):59–71, Jan 2012.

[64] R. Redon, S. Ishikawa, K. R. Fitch, L. Feuk, G. H. Perry, T. D. Andrews, H. Fiegler, M. H. Shapero, A. R. Carson, W. Chen, E. K. Cho, S. Dallaire, J. L. Freeman, J. R. Gonzalez, M. Gratacos, J. Huang, D. Kalaitzopoulos, D. Komura, J. R. MacDonald, C. R. Marshall, R. Mei, L. Montgomery, K. Nishimura, K. Okamura, F. Shen,

M. J. Somerville, J. Tchinda, A. Valsesia, C. Woodwark, F. Yang, J. Zhang, T. Zerjal, J. Zhang, L. Armengol, D. F. Conrad, X. Estivill, C. Tyler-Smith, N. P. Carter, H. Aburatani, C. Lee, K. W. Jones, S. W. Scherer, and M. E. Hurles. Global variation in copy number in the human genome. *Nature*, 444(7118):444–454, Nov 2006.

[65] T. Santarius, J. Shipley, D. Brewer, M. R. Stratton, and C. S. Cooper. A census of amplified and overexpressed human cancer genes. *Nat. Rev. Cancer*, 10:59–64, Jan 2010.

[66] J. Sebat, B. Lakshmi, D. Malhotra, J. Troge, C. Lese-Martin, T. Walsh, B. Yamrom, S. Yoon, A. Krasnitz, J. Kendall, A. Leotta, D. Pai, R. Zhang, Y. H. Lee, J. Hicks, S. J. Spence, A. T. Lee, K. Puura, T. Lehtimaki, D. Ledbetter, P. K. Gregersen, J. Bregman, J. S. Sutcliffe, V. Jobanputra, W. Chung, D. Warburton, M. C. King, D. Skuse, D. H. Geschwind, T. C. Gilliam, K. Ye, and M. Wigler. Strong association of de novo copy number mutations with autism. *Science*, 316(5823):445–449, Apr 2007.

[67] J. Sebat, B. Lakshmi, J. Troge, J. Alexander, J. Young, P. Lundin, S. Maner, H. Massa, M. Walker, M. Chi, N. Navin, R. Lucito, J. Healy, J. Hicks, K. Ye, A. Reiner, T. C. Gilliam, B. Trask, N. Patterson, A. Zetterberg, and M. Wigler. Large-scale copy number polymorphism in the human genome. *Science*, 305(5683):525–528, Jul 2004.

[68] S. Selvarajah, M. Yoshimoto, O. Ludkovski, P. C. Park, J. Bayani, P. Thorner, G. Maire, J. A. Squire, and M. Zielenska. Genomic signatures of chromosomal instability and osteosarcoma progression detected by high resolution array CGH and interphase FISH. *Cytogenet. Genome Res.*, 122(1):5–15, 2008.

[69] N. Shimizu, K. Shingaki, Y. Kaneko-Sasaguri, T. Hashizume, and T. Kanda. When, where and how the bridge breaks: anaphase bridge breakage plays a crucial role in gene amplification and HSR generation. *Exp. Cell Res.*, 302:233–243, Jan 2005.

[70] E. Shtivelman, B. Lifshitz, R. P. Gale, and E. Canaani. Fused transcript of abl and bcr genes in chronic myelogenous leukaemia. *Nature*, 315:550–554, 1985.

[71] J. Skarda, N. Amariglio, and G. Rechavi. RNA editing in human cancer: review. *APMIS*, 117:551–557, Aug 2009.

[72] P. J. Stephens, C. D. Greenman, B. Fu, F. Yang, G. R. Bignell, L. J. Mudie, E. D. Pleasance, K. W. Lau, D. Beare, L. A. Stebbings, S. McLaren, M. L. Lin, D. J. McBride, I. Varela, S. Nik-Zainal, C. Leroy, M. Jia, A. Menzies, A. P. Butler, J. W. Teague, M. A. Quail, J. Burton, H. Swerdlow, N. P. Carter, L. A. Morsberger, C. Iacobuzio-Donahue, G. A. Follows, A. R. Green, A. M. Flanagan, M. R. Stratton, P. A. Futreal, and P. J. Campbell. Massive genomic rearrangement acquired

in a single catastrophic event during cancer development. *Cell*, 144(1):27–40, Jan 2011.

[73] R. E. Sutton and J. C. Boothroyd. Evidence for trans splicing in trypanosomes. *Cell*, 47:527–535, Nov 1986.

[74] G. W. Tam, R. Redon, N. P. Carter, and S. G. Grant. The role of DNA copy number variation in schizophrenia. *Biol. Psychiatry*, 66(11):1005–1012, Dec 2009.

[75] G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences*, 65(3):587–609, 2002.

[76] S. A. Tomlins, D. R. Rhodes, S. Perner, S. M. Dhanasekaran, R. Mehra, X. W. Sun, S. Varambally, X. Cao, J. Tchinda, R. Kuefer, C. Lee, J. E. Montie, R. B. Shah, K. J. Pienta, M. A. Rubin, and A. M. Chinnaiyan. Recurrent fusion of TMPRSS2 and ETS transcription factor genes in prostate cancer. *Science*, 310:644–648, Oct 2005.

[77] C. Trapnell, L. Pachter, and S. L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25:1105–1111, May 2009.

[78] T. Walsh, J. M. McClellan, S. E. McCarthy, A. M. Addington, S. B. Pierce, G. M. Cooper, A. S. Nord, M. Kusenda, D. Malhotra, A. Bhandari, S. M. Stray, C. F. Rippey, P. Roccanova, V. Makarov, B. Lakshmi, R. L. Findling, L. Sikich, T. Stromberg, B. Merriman, N. Gogtay, P. Butler, K. Eckstrand, L. Noory, P. Gochman, R. Long, Z. Chen, S. Davis, C. Baker, E. E. Eichler, P. S. Meltzer, S. F. Nelson, A. B. Singleton, M. K. Lee, J. L. Rapoport, M. C. King, and J. Sebat. Rare structural variants disrupt multiple genes in neurodevelopmental pathways in schizophrenia. *Science*, 320(5875):539–543, Apr 2008.

[79] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10:57–63, Jan 2009.

[80] Wellcome Trust Sanger Institute. Cancer Gene Census. `http://cancer.sanger.ac.uk/cancergenome/projects/census/`, March 2013.

[81] J. Yu, J. Yu, R. S. Mani, Q. Cao, C. J. Brenner, X. Cao, X. Wang, L. Wu, J. Li, M. Hu, Y. Gong, H. Cheng, B. Laxman, A. Vellaichamy, S. Shankar, Y. Li, S. M. Dhanasekaran, R. Morey, T. Barrette, R. J. Lonigro, S. A. Tomlins, S. Varambally, Z. S. Qin, and A. M. Chinnaiyan. An integrated network of androgen receptor, polycomb, and TMPRSS2-ERG gene fusions in prostate cancer progression. *Cancer Cell*, 17:443–454, May 2010.

[82] J. Yu, J. Yu, R. S. Mani, Q. Cao, C. J. Brenner, X. Cao, X. Wang, L. Wu, J. Li, M. Hu, Y. Gong, H. Cheng, B. Laxman, A. Vellaichamy, S. Shankar, Y. Li, S. M. Dhanasekaran, R. Morey, T. Barrette, R. J. Lonigro, S. A. Tomlins, S. Varambally,

Z. S. Qin, and A. M. Chinnaiyan. An integrated network of androgen receptor, polycomb, and TMPRSS2-ERG gene fusions in prostate cancer progression. *Cancer Cell*, 17:443–454, May 2010.

[83] I. G. Yulug, A. Yulug, and E. M. Fisher. The frequency and position of Alu repeats in cDNAs, as determined by database searching. *Genomics*, 27:544–548, Jun 1995.