ALGORITHMS FOR STRUCTURAL VARIATION DISCOVERY AND PROTEIN-PROTEIN INTERACTION PREDICTION

by

Iman Hajirasouliha

M.Sc., Simon Fraser University, 2007

B.Sc., Sharif University of Technology, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2012

APPROVAL

Name: Iman Hajirasouliha

Degree: Doctor of Philosophy

Title of thesis: Algorithms for structural variation discovery and protein-protein interaction prediction

Examining Committee:

Dr. Torsten Moller
Chair

Dr. S. Cenk Sahinalp, Senior Supervisor
Computing Science, Simon Fraser University

Dr. Petra Berenbrink, Supervisor
Computing Science, Simon Fraser University

Dr. Inanc Birol, Supervisor
Bioinformatics group leader, GSC, BC Cancer Agency

Dr. Martin Ester, SFU Examiner
Computing Science, Simon Fraser University

Dr. Eran Halperin, External Examiner
Computer Science, Tel-Aviv University

Date Approved: August 1, 2012

Partial Copyright Licence

Abstract

This thesis has two main parts. In the first part, we will give an introduction on human genomic sequences, next-generation sequencing technologies, the structural differences among genomes of different individuals, and the 1000 Genomes Project. We will then discuss the problems of finding novel sequence insertions and mobile element insertions (e.g. Alu elements) in sequenced genomes. To identify those genomic variations with much higher accuracy than what is currently possible, we propose to move from the current model of (1) detecting genomic variations in individual next-generation sequenced (NGS) donor genomes independently, and (2) checking whether two or more donor genomes, indeed, agree or disagree on the variationswe will call this model the independent structural variation detection and merging (ISV&M) framework. As an alternative, we propose a new model in which genomic variation is detected among multiple genomes simultaneously.

The second part of the thesis focuses on a different project which is concerned with gene tree alignment. The aim is to present the first efficient approach to the problem of determining the interaction partners among protein/domain families. This is a hard computational problem, in particular in the presence of paralogous proteins. We devise a deterministic algorithm which directly maximizes the similarity between two leaf labeled trees with edge lengths, obtaining a score-optimal alignment of the two trees in question.

To Mom, Dad and Nima.

"I'm not going to tell the story the way it happened. I'm going to tell it the way I remember it."

Acknowledgments

I would like to express my utmost gratitude to my senior supervisor, Cenk Sahinalp. His continuous and unconditional support made my Ph.D. work and studies at SFU extremely joyful and exciting. I learnt a lot from Cenk during my time at SFU and the time we were visiting schools in Turkey on his sabbatical leave. Discussions with him not only shaped my research directions and career path, but also taught me many useful things at a higher schema. I feel incredibly indebted to him.

I have also been very fortunate to work with Evan Eichler and his group during my visits to University of Washington, and remotely from Vancouver. Evan and his group introduced me to many important biological problems, and helped me focus on Next Generation Sequencing problems. A large fraction of the results in the first part of this thesis was obtained through collaborations with them. Among the people in that group, I would like to especially thank Can Alkan for his continuing help and support.

Many thanks to Inanc Birol and Petra Berenbrink, my co-supervisors here at SFU for their great help and numerous insights on different research problems throughout my studies. I would also like to thank Noga Alon, Colin Collins, Ben Raphael, Anna Lapuk, Stanislav Volik, Sohrab Shah, Jeffrey M. Kidd, Peter Unrau, Fiona Brinkman, Martin Ester, Funda Ergun, Jens Stoye, Alexander Schonhuth and Alfonso Valencia, some of the great researchers and scientists whom I had the chance to talk or collaborate with, during my Ph.D studies. I have benefited a lot from their enlightening insights and their help.

I was also very grateful to work with a number of great fellow lab members on various projects. I would especially like to thank Fereydoun Hormozdiari for his great help and enthusiasm. Also, many thanks to Phuong Dao, Faraz Hach, Rahele Salari, Andrew McPherson, Emre Karakoc, Deniz Yorukoglu, Marzieh Bakhshi, Ibrahim Numanagic, Farhad Hormozdiari, Lucas Swanson, Reza Shahidi-Nejad, Kendrik Wang, and Yen-Yi Lin.

I would also like to thank my dear friends, Majid Bagheri, Lisa Brunner, Bradley Coleman,

Hossein Jowhari, Shilo StCyr, Murray Patterson, Robabe Sadr Eshkevari, Kamyar Khodamoradi, Hamed Yaghoubi, Mohammad Tayebi, Hassan Khosravi, Sara Saghaei, Bamdad Hosseini, Alireza Ghane, and Amir Hedayaty.

I should thank Eran Halperin for kindly serving as my external examiner, and also Vineet Bafna who accepted to be an examiner initially.

I acknowledge Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting my Ph.D. studies with an NSERC Alexander Graham Bell Canada Graduate Scholarships (CGS-D) and a Michael Smith Foreign Study Supplement scholarship.

Finally, I thank my family for the love and support they gave me in all these years. This thesis is dedicated to them.

Contents

List of Tables

List of Figures

Preface

High throughput - next generation sequencing (NGS) technologies are reducing the cost and increasing the world-wide capacity for sequence production at an unprecedented rate. These sequencing technologies have significantly changed how genomics research is conducted. In less than a decade from the completion of the Human Genome Project, the efficiency of DNA sequencing has increased by 100,000-fold and large scale projects based on sequencing of 2000 [5] or even 10000 individual genomes [31] are now possible to study. Although the current next-generation sequencing technologies offer a very high throughout with an effective cost, they produce much shorter reads compared to the traditional Sanger sequencing. The NGS reads are typically between 50 to 150bp, and this makes the analysis of the sequenced genomes very challenging. Genome sequence analysis and in particular the problems of identifying differences (*variations*) among individual genomes become much harder when dealing with short reads, mainly due to the complex, and repetitive nature of human genomes.

A fraction of genomic sequences translates to functional proteins. The vast majority of cellular functions are exerted by combinations of interacting proteins and it is a well-known fact that isolated cellular proteins are less likely to be functional. As a result, ”preservation of functionality” among proteins and other gene products typically implies ”preservation of interactions” across species. A major implication of this is that interacting proteins have a tendency to co-evolve, and the evolutionary trees behind two interacting protein families can look near-identical. Thus, assessing of two or more proteins being interaction partners can be done by measuring how similarly they evolve across related species. Understanding and accurate representations of interacting protein parteners and protein-protein interaction networks in general contribute to our knowledge of biological functions, as well as pathological states.

This thesis has two integral parts. The first part is focused on genome sequence analysis, in particular development of computational methods for structural variation discovery in sequenced

genomes. We present our effort in developing novel algorithms for identifying deletions, novel sequence insertions, transposon insertions, as well as a new combinatorial framework for handling structural variation discovery in multiple sample genomes. In this part, we also address an experimental design problem of re-sequencing experiments using the Illumina technology, based on bacterial artificial chromosome (BAC) clones. The second part of the thesis is, however, focused on computational problems related to protein interactions and their network. We first present an efficient approach to the problem of determining the interaction partners among protein/domain families, in particular in the presence of paralogous proteins. We also discuss topological features of protein-protein intraction (PPI) networks and present algorithms for comparing PPI networks among various organisms. Note that, while both parts of the thesis have strong connection to each other from a methodological point of view, each part investigates different biological problems.

Chapter one is an introduction on genome sequence analysis, which includes an overview of the human genome, next generation sequencing technologies and short read genomic data. In this chapter, we also discuss available methods for aligning short reads to the reference genome; typically the first step of the analysis of sequenced genomes. Later in this chapter, we talk about structural differences among individual genomes and present some of the methods and techniques for detecting these differences using short read data with a special focus on Variation Hunter, a combinatorial method developed earlier in our group. **Chapter two** is focused on a problem related to efficient use of next-generation sequencing technologies. Here our aim is to minimize the overlaps between BAC clones (pieces of a human genome) to be sequenced through the Illumina technology (i.e. one of the next generation sequencing technologies) so as to minimize potential genome sequence assembly errors. This work was presented at the ISMB 2008 conference and was published in [47]. **Chapter three** is the presentation of our effort in developing new methods for identifying novel sequence insertions and transposon insertions, using next generatrion sequencing data. In this chapter, we also present comprehesive experimental results on several whole genome sequencing data sets. We first presented these results at ISMB 2010 and ISMB-SIG 2010 conferences. This work was also published in [48, 57, 56]. **Chapter four** is focused on the *CommonLAW* package, our recent development in detecting structural differences among multiple samples simeutaniously. We also present the result of testing CommonLAW on high-coverage whole genome sequenced mother-father-child trios, which concludes the first part of this thesis. I presented this work at the RECOMB 2011, while the full paper was published in [58].

The second part of the thesis begins with **Chapter five**, where we introduce computational methods for identifying protein interaction parteners. In this chapter, we present our recent work in

designing a novel and efficient algorithm based on the similarities between phylogenetic profile of potential interactive partners. This work was published in [49]. **Chapter six** discribes a technique for enumerating and counting specific subnetworks in protein-protein interaction networks, and uses that technique to compare protein-protein intraction networks of several organism. We will conclude each of these chapters by discussing future directions and related open problems. I presented the ealier version of this work at the ISMB 2008 conference. This work was published in [10].

Chapter 1

Next generation-genome sequencing

Recent years have witnessed an increase in research activity in analysis of human genomes. With the introduction of next-generation sequencing (NGS) technologies, many groups have conducted extensive research on detecting genomic variations and their association to human disease. The advent of next-generation sequencing technologies make it possible to extend the scope of these studies to a point previously unimaginable as exemplied by the 1000 Genomes Project [5] and other large scale genome sequencing projects. Thousands of more individual genomes are now being sequenced in different institutions and genome centers around the world, including samples from patients suffering from diseases of genomic origins. Individual human genomes are different. Genomic variations not only include single nucleotide polymorphisms (variants at the single base pair level) or small indels (insertions or deletions of a few base pairs upto 50bps), but also larger and more complex events. According to one definition, events of size greater than 50bp are called structural variations [5]. Several types of structural variation events have been observed in human genomes; events such as deletions, insertions (e.g. novel sequence insertions), inversions, transpositions (e.g. Alu element insertions), translocations, or segmental duplications (i.e. duplicated sequences of size greater than 1000bp with sequence similarities of over 90%). The International HapMap Project genotyped 270 human individuals for 3.1 million SNPs [62, 63], and recently the 1000 Genomes Project[1] characterized human genetic variation with lower minor allele frequency by sequencing more than 1000 human genomes. In the first published result of the 1000 Genomes Project [5, 124], 15 million SNPs, 1 million short indels and 20,000 structural variants (SVs) were reported in 185 different individual genomes. Structural variation (SV) events significantly contribute to human genome diversity

[1] http://www.1000genomes.org

[147, 39, 81, 75, 99, 139, 5, 124]. Many structural variants are associated with genetic diseases such as psoriasis [52] and Crohn's disease [98], prompting an increased interest in structural variation studies in recent years [147, 81, 55, 16, 80, 29, 136, 87, 48, 57, 58, 124]. Structural variation detection through NGS promises to be one of the key diagnostic tools for cancer and other diseases with genomic origins [39, 138]. One recent study [85] for example, demonstrates that patient-specific structural variants identified in blood samples could be used as personalized biomarkers for monitoring tumor progression and responses to cancer therapies. The main potential use of NGS in clinical applications, however, would be the identification of genomic variants including the structural ones as recurrent biomarkers in patient subgroups which are scarcely observed in healthy tissues. Some recent studies on specific cancer types, on the other hand, have not been able to identify recurrent structural biomarkers (e.g. [30] or [95]). Although it is possible that such genomic signals simply do not exist in the cancer types studied, a more likely explanation is that the computational tools used in these studies were not sufficiently accurate to correctly identify and/or prioritize recurrent structural variants. Thus, it is possible that the road from personalized genomics to personalized medicine has been rough, partially due to the lack of sufficiently accurate computational tools for identifying recurrent structural variants among a collection of genomes and transcriptomes. Since the introduction of the paired-end sequencing of genome fragments, many research groups developed methods to detect and characterize different types of structural variations using next generation sequencing. For some of those early work on this topic, please see [81, 75, 86, 16, 55]. Note that all these works are based on mapping paired-end sequences onto a reference genome, which was introduced a few years earlier in [149, 122, 147]. The focus of the first part of this thesis is algorithmic methods for identifying structural variation events. In particular, chapter 3, which is the lengthiest and most involved chapter of the thesis, focuses on two novel frameworks for detecting two classes of structural variations; novel sequence insertions and transposon insertions. Less methods were developed to identify these two classes of structural variation events recently, due to the challenges we will discuss in chapter 3. In the current chapter, we first give an introduction to next generation sequencing and then present some of the existing powerful algorithms for mapping short reads to a reference genome. Next, we describe some of the recent techniques for structural variation discovery in sequenced genomes, in more detail. We discuss the main ideas currently being employed in these methods, and present the strengths of each method. We also argue that ignoring repeat regions in the genome is not simply an answer to important biological question and focus on methods which aim to handle repeat regions.

Next-generation sequenced human genomes The human genome makes up 23 choromosome pairs and the haploid genome is estimated to have 3 billions base pairs (bp). Each base pair is from the alphabet A, C, G, T (the first letters of the words adenine, cytosine, guanine, and thymine), which are the four nucleobases in the nucleic acid of DNA. In 2003, after several years of work on a large international consortium with a budget of 3 billion dollars, a (near) complete reference genome as the result of the human genome project became available. The human reference genome contains sequences mainly from a few individuals. Since then, researchers from all over the world were able to download and use the reference genome for their studies. Parallel to the public effort by the human genome project, Celera Corporation (a company led by Craig Venter) aimed to build a complete reference genome from a pool of individual genomes which included Venter's own genome. Later in 2007, Venter's group published the first complete genome of an individual human, his own genome [89]. The original human genome project generated approximately 30 million reads (500–1000 bps per read) via Sanger sequencing. Currently, with the advent of the sequencing technologies, as many as 6 billion reads (50-150bps per read) per run can be produced [146]. The recent Illumina HiSeq technology is now being used widely and is capable of producing 2×100bp paired-end reads in its High Output mode, and $2x150$bp reads in its rapid mode [3]. Earlier Illumina platforms such as Illumina Genome Analyzer are still being used by the sequencing centers around the globe. Figure 1 shows the fragment size distribution of paired-end reads of a Yuroba genome which was sequenced by the Illumina Genome Analyzer platform. Note that Illumina offers several thousand-fold improvement over Sanger sequencing in terms of cost and the cost is now becoming feasible for many biological and medical applications. Similarly, the pyrosequencing technology of 454 Life Sciences [96] uses massive parallelization of the sequencing process by the use of microchip sensors, producing single reads of size approximately 400bp. Figure 1 shows an example of read length distribution from a 454 GS Junior shotgun sequencing run (E. coli K-12). [1]. ABI solid is another next generation sequencing technology which produces short color-space reads [2]. The focus of our report is on data generated by Illumina platforms.

The 1000 Genomes Project An example of a large scale genome analysis project using massive sequencing data is the 1000 genomes project. The aim of the project is to sequence and create a fine scale map of genetic variation of 2500 unidentified individual genomes from 25 different populations. Similar to the human genome project, the sequences and the results of this study will be freely and publicly accessible to the community. In the pilot 1000 Genomes Project, 179 unrelated

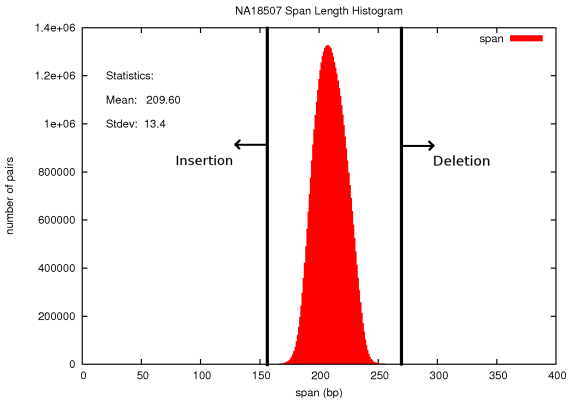


Figure 1.1: An example of paired-end fragment size distribution of the sequenced genome of NA18507, a Yoruba individual, by the Illumina Genome Analyzer platform.

individuals from diverse populations were sequenced to a low coverage using next-generation sequencing technologies. In addition, in a study known as the trio project, the genomes of a Yoruba (from Ibadan, Nigeria) parent-offspring trio and a trio of European ancestry from Utah were each sequenced to 30X coverage [5]. The pilot project generated a population-scale sequencing data set (i.e. more than 4.1 Terabases of raw sequences) and reported the location and allele frequency of approximately 15 million SNPs, 1 million short insertions and deletions and 20,000 structural variants, the majority of which were previously undescribed [5]. In a separate Nature publication [124], the results of the 1000 Genomes Project Structural Variation subgroup was presented. In [124], the authors presented the discovery and validation of structural variation events of greater than 50bp. They also compared different discovery approaches using sequence data generated from both whole genome pilot projects and developed a fine-scale map of this variation at the breakpoint level. The focus of the study was initially on deletions, an SV class often associated with disease and for which rich control datasets as well as diverse ascertainment approaches exist. Less focus then was placed on insertions (including novel sequence insertions) and duplications. Specifically, in this study 44 SV discovery callsets (23 deletion, 11 insertion, and 10 duplication callsets) were generated. This study showed that distinct SV discovery approaches ascertain SVs in a highly complementary fashion, enabling us to exploit evidence from different approaches in an SV analysis framework for constructing a comprehensive SV discovery set.

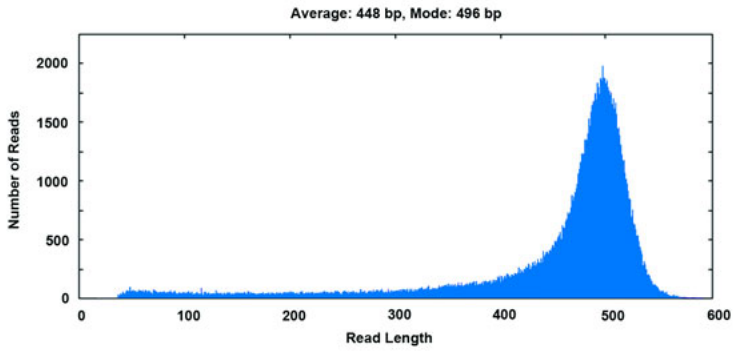


Figure 1.2: An example of read lengths of a sequenced E. coli K-12 produced by the 454 platform as was shown in the company website (`http://www.454.com/`). In this example, the average read length is 448 bp with a mode read length of 496 bp.

1.1 Mapping short reads onto a reference genome

In order to analyse high throughput sequenced genomes, the first step is usually to align the short read data onto a reference genome. Given the large amount of data, it is very desirable to have mapping methods that are not only reliable but also efficient in practice. In what follows, we discuss a few popular mapping algorithms for short read data that are currently being used in genomic applications.

BWA, Bowtie, and Bowtie2 The BWA [90] mapper works based on backward searching with Burrows-Wheeler Transform (BWT) and is designed to align short reads onto a reference genome. BWA is efficient and allows gapped alignments as well as mismatches. Note that the earlier alignment algorithms such as MAQ [91] were not able to handle gapped alignments and thus were not scalable to the newer generation of reads, which are longer. Gapped alignments are particularly important for aligning longer reads onto a reference genome because the probability of a short indel is higher in those reads. Furthermore, BWA is more than 10-fold faster than MAQ, while achieving similar accuracy [90]. The speed is a key issue when dealing with resequencing of hundreds of individuals. Bowtie [84] and Bowtie2 [83] are other fast and popular alignment tools widely being used by the community. Bowtie also indexes the genome with a Burrows-Wheeler index for memory purposes; however, it employs additional heuristics to be executed fast while maintaining a high sensivity. Note that Bowtie is an ungapped aligner and is the basis for other tools for analysing RNA-seq such as TopHat [145]: a splice junction mapper for RNA-seq reads, and Cufflinks [125],

a tool for isoform quantization and transcriptome assembly from RNA-seq reads. Bowtie2 [83] is a very recent improvement over Bowtie which allows gapped alignments, particularly good for aligning Illumina reads of size about 50bp to 100bp. Bowtie2 also uses an index-based approach similar to Bowtie but employs dynamic programming and benefits from the efficiency of single-instruction multiple-data (SIMD) of current processors. [83]. Note that earlier developments of short read aligners such as mrFAST [7] also used SIMD instructions for optimization purposes. mrFAST [7] and its follow-up mrsFAST [46] were aligners designed in our group to map short reads generated with the Illumina platform to reference genome assemblies in a fast and memory-efficient manner. These aligners are the first step of our Structural Variation detection pipeline [55, 48, 57, 58] and, in addition to the standard SAM format, produce output formats suitable for VariationHunter [55] and NovelSeq [48], two of our pipelines. In the next section, we describe the core method used in mrFAST and mrsFAST.

mrFAST and mrsFAST mrFAST, a micro-read Fast Alignment Search Tool, is an efficient gapped aligner that reports all possible alignment locations of a read onto a limited number of mismatches and indels (i.e. small insertions and deletions). It currently supports indels up to 8 bp (4 bp deletions and 4 bp insertions) and outputs all possible alignments of the paired-end reads in the SAM format. mrFAST was initially designed to count duplicated regions in personal genomes. The additional features such as *discordant* and *One end anchored (OEA)* mapping options make it easier to use mrFAST for structural variation discovery tools such as VariationHunter [55] and NovelSeq [48]. mrFAST is a seed and extend method and uses a collision-free hash table to index the reference genome. Due to main memory constraints, mrFAST is not able to store the whole human genome index and thus partitions the reference genome into smaller contigs; first partitioning along the reference sequence gaps. Each ℓ-mer of the genome ($\ell \approx 12$) is stored in the hash table. When mrFAST searches for the possible alignment location of a short read in the *seed* step, exact matches of the seed will be selected; this step can be done very quickly using the hash table. In the *extend* step, a dynamic programming approach based on the Smith-Waterman algorithm is used to find mapping locations of each paired-end read with a high sequence similarity, allowing a small edit distance. For additional optimization, mrFAST utilizes the SSE2 instruction set extensions available in modern CPUs. Using SSE2, multiple values of the dynamic programming table can be found with only a single instruction [7]. mrsFAST is a followup of the mrFAST algorithm. This short read aligner does not allow indels (i.e. only mismatches are allowed) and thus executes faster than mrFAST. Also, mrsFAST is a short read mapper, which means optimizes cache usage to get a higher

	Read Length: 50 bp (3 errors)			Read Length: 75 bp (4 errors)		
	Time	(%) Mapped	Mappings (10^6)	Time	(%) Mapped	Mappings (10^6)
Bowtie	3:13	92.73	610	NA	NA	NA
BWA	10:23	93.38	729	59:35:00	90.16	212
Maq	10:05	89.25	458	NA	NA	NA
mrFAST-CO	9:21	93.39	663	11:32	90.22	193
mrsFAST	1:55	92.91	613	2:00	89.35	177
BWA*	0:15	93.38	< 1	0:25	90.16	< 1

Table 1.1: The runtime (h:min), percentage of reads, and number of mapping locations reported (millions) by each aligner method. Note that, in this experience, 1 million paired-end reads were mapped to the human build36. BWA* only reports a single mapping location.

performance. This aligner indexes both the reference genome and the reads, and executes a simple cache oblivious, all-to-all list comparison algorithm [46]. mrsFAST minimizes the total number of cache misses. Table 1.1 is from part of a table presented in the mrsFAST paper [46] which compares the performance of different alignment methods. It shows how different methods perform when short reads of size 50bp and 75bp are aligned to the human reference genome.

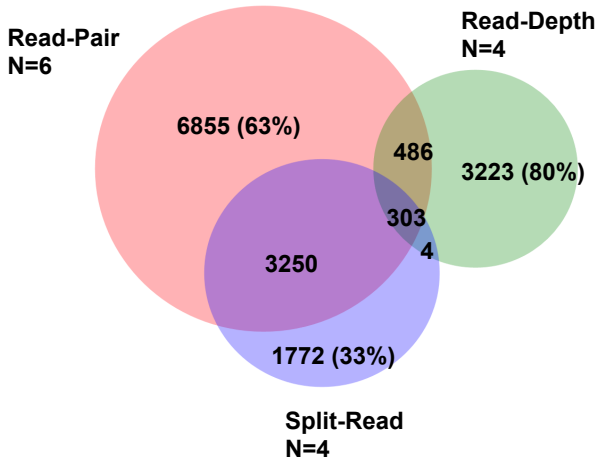


Figure 1.3: Validated deletions from the 1000 Genomes pilot data [124]. As it can be seen no method is comprehensive. This figure was presented in [6].

The main developers of mr and mrsFAST also developed drFAST [54], an aligner for SOLiD color space reads as well mrFAST-CO [46], which is the cache oblivious extension of mrFAST. Other popular alignment tools, developed for aligning longer contigs, which used seed and extend

approach include BLAST [12] and BLAT [74].

1.2 Methods for Structural variation discovery and their limitations

In this section, we give a quick overview of the current structural variation (SV) discovery methods with a special focus on VariationHunter [55]. Traditionally, SV discovery methods are classified into four major categories, based on the general approach they use. The approach used in each category is significantly different from other SV discovery approaches. In this section, we discuss these categories. Figure 1.3 shows the common and specific deletion events found by methods from different categories of the 1000 Genomes Project. As it can be seen, none of the methods are comprehensive. Some groups have recently started working on integrative methods (i.e. discovery methods which utilize signatures from two or more categories in order to achieve improvements). At the end of this section, we review the recent integrative methods as well.

- Read Pair (RP) methods: use paired-end sequencing and detect structural variations by the means of mapping paired-end reads to a reference genome and observing discordant mappings.
- Split Read (SP) methods: typically use gapped alignments of single reads to pinpoint the (near) exact breakpoint of structural variation events. These methods can handle small insertion and deletion events as well.
- Read Depth (RD) methods: use the signature of number of mapped short read in each window of the reference genome to identify duplication regions or deletion regions in a sequenced genome.
- Assembly (AS) methods: rely on de novo assembly of the sequenced genomes and compare the assembled contigs against the reference to identify indels and structural variation events.
- Integrative methods: use sequence signatures from two or more of the above categories (e.g. RD and RP)

In the following subsections, we discuss the power and limitations of each of the above structural variation discovery methods, except the Read Depth methods, with a special focus on Read Pair methods. We also highlight some of the popular methods, in each subsection.

1.3 Read Pair methods

A large fraction of the available methods for SV discovery employ paired-end sequencing: inserts from a donor genome (from a tightly controlled length distribution) are read at two ends, which are later aligned to a reference genome. In [147, 149], a general framework for detecting structural variation using long paired-end reads was introduced. This framework is based on aligning the paired-end reads to the reference genome and observing the end-reads (i.e. the two ends of a read pair) with discordant mapping. The paired-end reads with discordant mapping suggest either insertion or deletion events or more complex events such as inversion, translocation, transposition, and duplication. For example, an inversion event can be deduced when one of the two end-reads of a paired-end read has a different mapping orientation than expected. In the standard library construction of the Illumina platform, in the case of no inversions or duplications, the read that maps to the proximal location is expected to be in the $+$ strand, where its mate should be mapped to a distal location in the $-$ strand. However, if the read pair spans an inversion breakpoint, the mapping orientations of the reads will be observed as either $++$ or $--$. Figure 1.4 shows sequence signatures of different types of structural variation events. Later in this section, we will give a case study for different events.

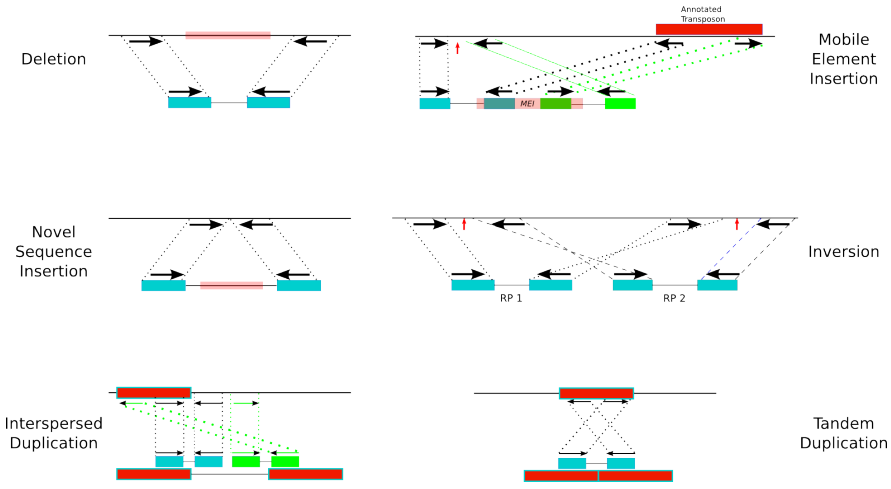


Figure 1.4: Signatures for different types of structural variation event such as Deletion, Transposon Insertion, Novel Sequence Insertion, Inversion, Interspersed Duplication and Tandem Duplication are presented in this figure.

With the advent of next-generation sequencing technologies, many groups used this general framework to identify structural variation in high throughput sequenced genomes. PEmer [80] for

example, maps each paired-end read to a unique location through the mapping software MAQ [91]. A number of followup studies that employ a similar "hard clustering" approach [105] including Pindel [158] and BreakDancer [29], all focus only on the "best mapping" of each read, provided by the mapping software in use. A survey by Medvedev et al.[105] summarizes the basis of decision making for each of these methods and reports briefly on their performance. A more comprehensive survey by Alkan et al. [6] reviews the advantages and limitations of different approaches for genome structural variation discovery and genotyping. These methods typically work well on unique regions of the human genome. However, they naturally ignore potential multiple alignment locations in repeat regions by either picking one arbitrary location among many possibilities or simply avoiding the use of reads that have multiple mapping locations. As a result they cannot capture structural variations in repetitive regions of the human genome. In a paper by our group [46], it was demonstrated that ignoring possible mapping locations of a read may lead to significant loss of accuracy in structural variation detection. Other striking examples were shown in a recent survey by Treangen and Salzberg [146], proving that simply ignoring repeats is not an option. Ignoring repeats may mean that very important biological phenomena are missed [146]. A 2x100bp read provided by Illumina HiSeq2000 technology maps to more than 180 locations within 6 mismatches or indels. Picking an arbitrary location among these as the mapping location of a read naturally leads to both false positives and false negatives in SV discovery. To address the above problem, a number of "soft clustering" techniques [55, 7, 87] have been introduced in the past three years. Here, paired-end reads are mapped to all potential locations - through the use of the mapping algorithms such as mr and mrs *FAST* [7, 46] (See Section 1.1 for a more detailed description of these mapping algorithms.) In soft clustering approaches, paired-end reads can have multiple mapping to the reference genome, and thus suggest different variations. Each set of the discordant paired-end reads can be indicating a real structural variation or just be an artifact of the multiple mapping. These clusters of paired-end reads are denoted as soft-clusters [105]. VariationHunter [55] is one of those soft clustering methods that aims to resolve repetitive regions of the human genome through a combinatorial optimization framework for detecting insertion and deletion polymorphisms. Note that prior to the publication of VariationHunter [55], a probabilistic framework by Lee et al. [86] was presented for detecting structural variation. In their work a scoring function for each SV was defined, as a weighted sum of (1) sequence similarity, (2) length of SV and (3) the square of number of paired-end reads supporting the SV. The scoring function was computed via hill climbing strategy to assign paired-end reads to SVs [55]. The conducted experiments of [86] was based on capillary sequencing[89], but the framework could be applied to next generation sequenced genomes as well. In [57], a novel algorithm

for transposon insertion discovery was introduced, while a new computational pipeline, NovelSeq, for novel sequence insertion discovery was presented in [48]. Both of these methods employ soft clustering techniques. MoDiL [87], as well as its followup MoGUL [88] evaluate the clusters of reads that seem to indicate a structural variant using a probabilistic framework, while Hydra [119] uses heuristics (based on the algorithmic strategies of VariationHunter) to detect structural variant breakpoints in the mouse genome. MoGUL [88] focuses on finding common insertion and deletion events in a pool of multiple low coverage sequenced genomes.

In MoGUL [88] a method based on Bayesian networks is suggested to predict common small and medium size indels in a group of individuals sequenced in low coverage. Although the above strategies are quite useful in detecting SVs between a donor genome and a reference genome, they cannot be relied on when the goal is to discover SVs between two or more donor genomes. In the rest of this section, we first give a short summary of BreakDancer [29], a Read Pair method which uses best mappings (i.e. hard clustering). Next, we give a detailed presentation of the techniques used in VariationHunter [55]. We remind the reader that VariationHunter and its followup methods [55, 57, 48, 58] consider all ambiguous mappings in their core algorithms.

BreakDancer As we mentioned earlier, BreakDancer [29] utilizes only unique mappings produced by short read aligners such as MAQ [91] or BWA [90]. BreakDancer has two versions: BreakDancerMax (which was designed for detecting events of size larger than 100bp), and BreakDancerMini (which was designed for detecting events as small as 10bp and upto 100bp). In what follows, we briefly review BreakDancerMax. After filtering low quality mappings, BreakDancerMax classifies read pair alignments as: Normal, deletion, insertion, inversion, intra-translocation or inter-translocation. If an insert is not *normal*, it is called ARP (anomalous read pair) and an SV event is reported, if at least 2 ARPs are the same genomic location. BreakDancerMax also assigns confidence scores to each potential event. The probability of having more than the observed number of inserts in a particular genomic region is given by $P(n_i \geq k_i)$, where i is the type of insert, n_i is a Poisson random variable with mean equal to λ_i (we will shortly show how λ_i is estimated), and k_i is the number of observed type i inserts in the region. λ_i is estimated with $\frac{sN_i}{G}$, where s is the size of the region in which ARPs are anchored, N_i is the total number of ARPs of type i in the dataset, while G is the length of the reference genome. BreakDancer aims to find statistically significant SVs. i.e. those events with $p < 0.0001$

VariationHunter VariationHunter [55] presents combinatorial algorithms for structural variation detection using next generation sequencing. The authors of the paper introduced combinatorial formulations for SV discovery in high-throughput sequenced genomes using next generation data. Their formulations are based on the *Maximum Parsimony* principal. The algorithm aims to find the minimum number of structural variation events such that all mappings which are not *normal* can be explained. The authors modeled this problem with a combinatorial optimization problem based on *set cover*. While this optimization problem is NP-hard, a $O(\log n)$ approximation algorithm was given to solve this problem efficiently. Note that the initial implementation of VariationHunter was designed to identify small insertion events, medium size and large deletion events, and inversions [55]. Later, the VariationHunter method was extended to identify transposon insertions in sequenced genomes. [57]. Moreover, this work extends the maximum parsimony approach to non-conflicting maximum parsimony; a new formulation to discard potential events which have conflict with other structural variation events in the sequenced genome. At the core of the above general strategy is the computation of the expected distance between mate pairs in the donor genome, which is referred to as insert size ($InsSize$). Previous works [147, 81, 86] assume that for all paired-ends, $InsSize$ is in some range $[\Delta_{min}, \Delta_{max}]$ which can be calculated as described in [147]. An alignment of a paired-end read to reference genome is called *concordant* [147], if the distance between an aligned ends of a pair in the reference genome is in the range $[\Delta_{min}, \Delta_{max}]$, and both the orientation and the chromosome the paired-end read is aligned to are "correct". For instance in Illumina platform (for other platforms it might be different), a paired-end read is considered to be aligned in "correct" orientation if the left mate pair is mapped to the "+" strand (which is represented by +), and the right mate pair is mapped to the "-" strand (which is represented by -). A paired-end read which has no concordant alignment in reference genome as defined in [147] and later used in [81, 86], is called a *discordant* paired-end read (which indicates a possible structural variation). Note that this is similar to what BreakDancer [29] calls not *normal*. However, in VariationHunter, these discordant paired-end reads can have multiple locations in the genome that they can be aligned to with a high sequence similarity [55]. The algorithms in VariationHunter will obtain a unique alignment for each discordant paired-end read using a maximum parsimony approach. First, maximal sets of discordant paired-end read mappings such that all of the mappings in each set *support* the same structural variation event, are identified as *maximal valid clusters*. The Maximum Parsimony Structural Variation (MPSV) problem asks to compute a unique mapping for each discordant paired-end read in the reference genome such that the total number of implied structural variants (SVs) is minimized. It can be proved that the MPSV problem is NP-hard and an approximated solution exists, modeling

the problem with a set cover problem. For the details of the proofs, please see [55]. Note that a modification of the MPSV problem [55] for discovery of gene fusions using RNA-Seq data was implemented in [102], and later in [103, 104]. As we mentioned earlier, VariationHunter [55, 57] finds all maximal valid clusters for different types of structural variation events such as small insertions, deletions, inversions and transposon insertions. For each type, this can be done by an efficient technique to find all maximal intersecting intervals given a set of intervals in a one-dimensional space. For details please see [55, 57]. In what follows, we briefly present the clustering rules (i.e. breakpoints formula) for different types of structural variations. For full details and related algorithms, we refer the reader to [55, 57].

- Insertion: A cluster C is a valid cluster supporting insertion events if there exists a locus (breakpoint) in the reference genome where all of the paired-end read mappings in the cluster C span it. Also, the length of the insertion must obey a rule (i.e. must be small enough in this case). More formally:

$$\exists loc, \forall APE \in C : L(APE) < loc < R(APE) \tag{1.1}$$

$$\exists InsLen, \forall APE \in C : \Delta_{\min} - R(APE) + L(APE) < InsLen < \Delta_{\max} - R(APE) + L(APE). \tag{1.2}$$

  Where, in the above equations, $\Delta_{\min}, \Delta_{\max}$ refer to the minimum and maximum insert size of the paired-end reads. APE is a paired-end alignment, and $L(APE)$ and $R(APE)$ the left and right location of the alignment on the reference genome, respectively.

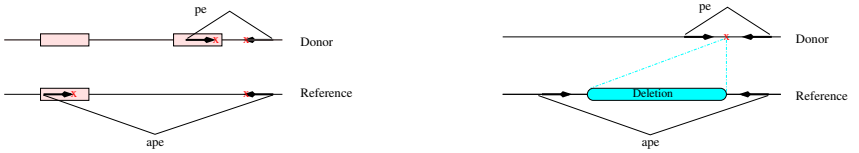
- Deletion: Deletion events have two breakpoints (denoted as Br_ℓ and Br_r), and VariationHuner aims to find those breakpoints approximately. Again, pair-end read alignments must obey certain rules in order to support a deletion event with breakpoints Br_ℓ and Br_r on the genome:

$$\Delta_{min} < R_\ell(APE) - Br_r + Br_\ell - L_r(APE) < \Delta_{max} \tag{1.3}$$

  Where $R_\ell(APE)$ and $L_r(APE)$ are mapping locations of the paired-end read APE on the genome. Again ,$\Delta_{\min}, \Delta_{\max}$ are the minimum and maximum insert size of the paired-end read, respectively.

- Transposon insertion: Another important type of structural variation is the *transposition* event, where a segment of the genome (formally, a transposon) is copied to another location with a small divergence. Examples of common transposon events include transpositions of Alu, SVA

and L1 elements. Most available methods designed to detect structural variation events (e.g. [147, 81, 75, 16, 86, 87, 55, 29, 136]) were not able to identify transposition events, and their focus was mainly on the discovery of deletions, insertions, and inversions. A more recent algorithm, HYDRA, includes simple heuristics to detect transposon insertions [119]. Interestingly, even if the goal of a method is to identify only insertions, deletions and inversions in a sequenced genome, the presence of the transposition events will cause many false negative deletion and inversion predictions. Thus, for read pair signature of transposon insertions, one has to be very careful since the signature can be easily mistaken for a deletion signature. Figure 1.5 shows an example that a signature for a transposition event can be identical to a deletion event:



(a) The presence of a transposon insertion in the donor genome and the supporting discordant paired-end mapping.

(b) The same discordant mapping as above, however, suggests a large deletion if the annotated repeat element is not considered.

Figure 1.5: Transposon insertion misleading to a false negative deletion prediction. A discordant paired-end read alignment due to a copy event shows identical pattern with a discordant paired-end read alignment supporting a deletion event.

In this thesis, we will study two classes of transposition insertions and present the set of conditions based on the map locations and orientations of the paired-end alignments. First, we will consider those events in which the transposed segment is in direct orientation, and present the set of conditions for all of the four different cases of this class (denoted as Class I). We denote this type of transposition event by $SV_{Copy}(Loc_L, Loc_R, Loc_{Br_L}, Loc_{Br_R})$. This indicates a region being copied is a segment inside loci $[Loc_L, Loc_R]$ (i.e. a substring of region $[Loc_L, Loc_R]$), and it is pasted (copied) to a locus between Loc_{Br_L} and Loc_{Br_R}. We will also study the cases for Class II, where the transposon is copied in inverted orientation, and we denote the event as $SV_{\overline{Copy}}(Loc_L, Loc_R, Loc_{Br_L}, Loc_{Br_R})$. In chapter 3, we discuss our formulation and methods for identifying transposition insertion, in more detail.

1.4 Assembly methods (AS)

Using de novo assembly of sequenced genomes is another way of identifying structural variation events. The basic idea is to collect *all* reads and use an available assembly software to assemble the read into contigs/scaffolds. By aligning larger contigs to the reference genome, structural variation events may be identified. Velvet [160], EULER [115, 27, 28], ABySS [134, 23], Cortex [64], SOAP-denovo [92], ALLPATHS-LG [25] are among notable de novo assembly algorithms. Note that most of these assembly algorithms are based on de Bruijn graphs. They further divide reads into k-mers, and build a de Bruijn graph on all the k-mers (i.e. for each k-mer, these is a node in the graph). Each two k-mers which share a suffix-prefix of size $k-1$ are connected via an edge [115]. After an error correction step, these methods use various heuristics to find correct Eulerian-type paths in the de Bruijn graph. Figure 1.6 shows the general framework of how assembled contigs help identifying structural variation events. For a summary of the popular de novo assembly software, and a comparison study we refer the reader to the recent paper, Assemblathon [36].

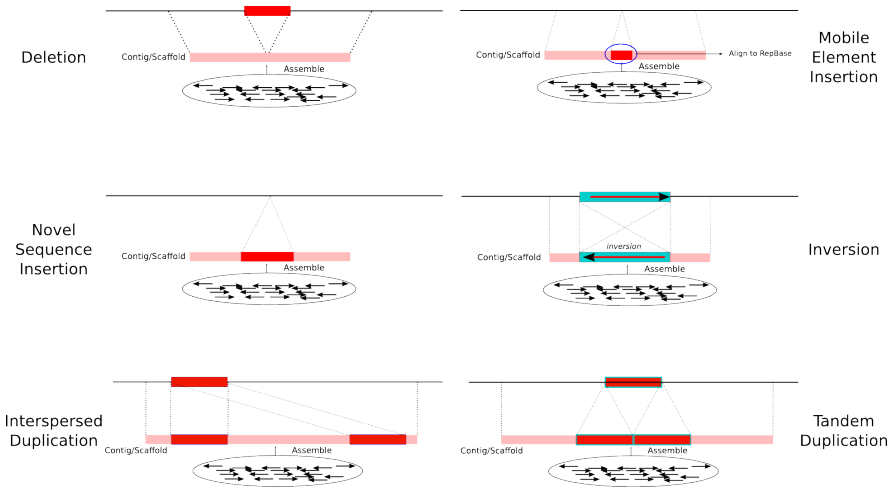


Figure 1.6: Signatures for different types of structural variation event such as Deletion, Transposon Insertion, Novel Sequence Insertion, Inversion, Interspersed Duplication and Tandem Duplication are presented in this figure.

Novelseq [48] Novelseq is a method that uses assembly to detect and characterize sequences which are present in sequenced genomes of study but missing in the reference. It is estimated that 19-40 Mb of human genomic sequence is missing from the human genome reference assembly [92]. Although

the Human Genome Project revolutionized the field of genomics, the human sequences not represented in the reference genome leads to incomplete genome analysis. The missing sequences can even harbour not-yet-discovered genes, or other types of sequences of functional importance. There is a need to discover the locus and content of so called "novel sequence insertions" to build more a comprehensive human reference genome to better analyze genomes of individuals from many different populations. To date, the most promising method to characterize longer DNA segments that are not represented in the human reference genome has been building sequence assemblies from unmapped fosmid clone ends sequenced with the traditional Sanger-based capillary sequencing [75], and sequencing the entire fosmid clones [77]. However, the higher cost of the capillary sequencing is prohibitive to characterize genomes of thousands of more individuals. Next generation sequencing technologies make sequencing of thousands of genomes possible, and for the first time, give us the opportunity to discover novel sequences across many human populations to build better genome assemblies (or pan genomes[92]). Various computational methods were developed in the recent years to characterize structural variation including deletions, insertions, inversions, and duplications among human individuals using next generation sequencing (NGS) platforms [105]. Characterization of locus and content of longer novel sequences remained elusive due to the shorter insert size and sequence length associated with the NGS methods. For example, using the end-sequence profiling approach [149, 147, 81, 75] one cannot discover insertions > 100 bp when 200 bp insert size is used with the Illumina platform [21, 55, 87, 29]. One applicable method for the discovery of long novel insertions using NGS technologies is *de novo* sequence assembly [134, 28, 160, 92]. However, this approach requires large computational resources, and requires further processing to anchor the sequences to the reference genome. Novelseq [48] utilizes de novo assembly together with signatures from paired-end read to find the content and location of novel sequence insertion in a sequenced genome.

1.5 Split Read methods

The lengths of the reads produced by sequencing technologies are becoming longer, and by the means of Split Read approaches, it is now possible to pinpoint exact breakpoints of indels and structural variation events, especially in unique regions. Figure 1.5 shows the general framework of the Split Read (SR) approaches. Pindel [158] is one of the first published papers, using split reads to identify structural variations. Pindel only allows unique mappings, and uses a *pattern growth* approach to search for unique substrings of unmapped reads in the genome. The algorithm then

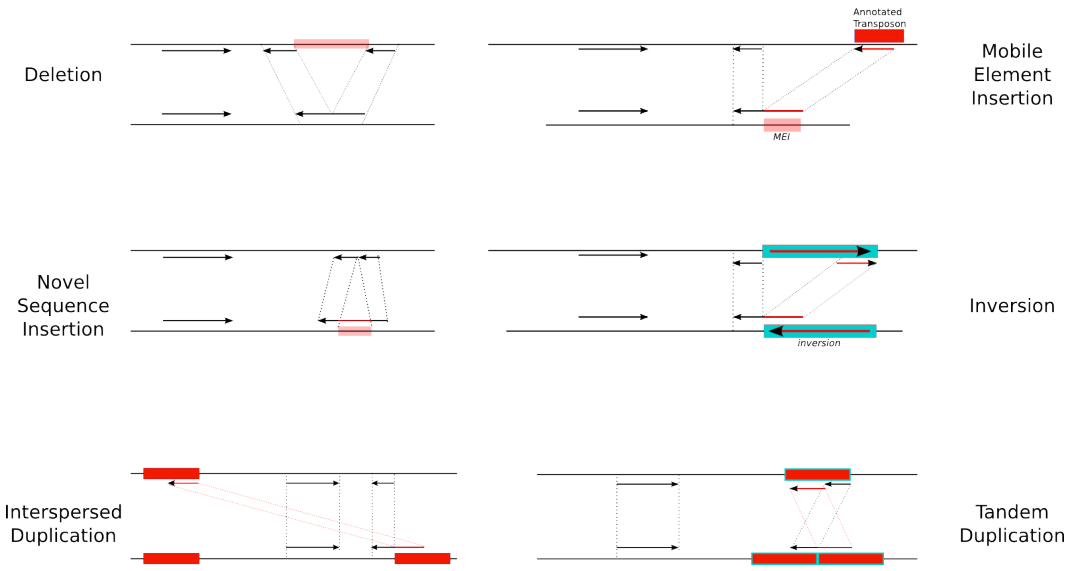


Figure 1.7: Split read analysis for different types of variants is shown in the figure. As it can be seen in the figure, using split reads helps pinpointing the breakpoints more precisely

checks whether a complete unmapped read can be reconstructed combining the unique substrings found in the previous step. SRiC [164] is a recent method designed for split read analysis of reads from the 454 platform. Note that 454 reads are typically about 400bp long, and thus it is easier to perform split read analysis on them, compared to Illumina reads. A very recent algorithm (i.e. SPLITREAD) by Karakoc et al. [70] utilizes multiple mappings in a split read analysis framework. The techniques used in [70], for clustering the alignments are based on the maximum parsimony approach which was introduced in [55]. We should mention here that TopHat[145], specialized for RNA alternative splicing, as well as Dissect [159] also uses split read alignments.

1.6 Integrative models

Due to the fact that none of the current methods for SV discovery in comprehensive, many researches have started to look into ways to integrate multiple signatures in one pipeline to improve the accuracy of SV discovery methods. Most of these integrative models combine signatures form discordant paired-end reads and read depth signatures. For example, SPANNER by Stewart et al. first finds candidate using read pair signatures, and then filter out some of those candidates using read depth signatures as a post processing step. Genome STRiP [51] uses both read pair and read depth signatures for structural variation discovery, similar to SPANNER. Moreover, Genome STRiP

integrates multiple genomes in a population for genotyping purposes. CNVer by Medvedev et al. [106] builds a graph using read pair signatures with weights obtained through read depth signatures. CNVer aims to find copy number variants by modeling the problem with a minimum cost flow problem in the graph. Finally and very recently, Sindi et al. [135] introduced an integrative probabilistic model that combines both read pair and read depth signatures. Their method, GASVPro, is based on their earlier structural variation detection work [136] and uses soft clustering to handle multiple mappings of reads in repeat regions.

Chapter 2

Optimal pooling for a genome re-sequencing experiment

As we discussed in the previous chapter, after decades of research effort, the cost of sequencing an individual human genome via Sanger sequencing [129] has now been reduced several orders of magnitute. For example, the Illumina technology offers more than 1000-fold improvements over Sanger sequencing in both cost and throughput. Similarly, the pyrosequencing technology of 454 Life Sciences [96] delivers massive parallelization of the sequencing process by the use of microchip sensors, improving the speed of Sanger sequencing, significantly.

In addition to these, there are other commercial products such as SOLiD [2] perform either clonal cluster sequencing, or single molecule sequencing.

Unfortunately, the massive increase in the throughput offered by the above technologies comes with a shortened read length, and shorter the read length, the more problematic it is to work with a genome that has many longer repeats. While Sanger sequencing offers $500 - -1000$bps per read, the read lengths of new technologies range from 36 to 150 (e.g. Illumina) to a few hundred base pairs (e.g. 454). Although in practice most sequencing technologies can produce longer, as well as paired end reads, but for the rest of this chapter we concentrate on the problem of resequencing of genomes with short single end reads, and focus only on the Illumina technology.

The type of re-sequencing problem we consider in here is based on bacterial artificial chromosome (BAC) libraries. Benefits of using BACs in re-sequencing studies is 2-fold. (1) It is possible to cluster the short reads obtained from each BAC into small local sets that represent the sequence of one BAC where most of the repeat sequences have a unique copy. (2) One can a priori determine the

genomic neighbourhood the BAC is coming from; thus having the reference sequence provides an essential backbone. This can be achieved, for instance, by using the results of fingerprinting or BAC end sequencing experiments. Although on the face of it, a BAC-library-based sequencing effort defeats the benefits of massive parallelization offered by new sequencing technologies, it enables directed sequencing studies possible, where the interest is not on whole genome (re)sequencing, but on investigating a region of interest. Furthermore, recent developments bring the throughput in fingerprinting technology on par with new sequencing technologies [97].

Here we look at the problem of how to design resequencing experiments to decrease the complexity of the downstream analysis. Resequencing is defined as using sequencing technologies to identify sequence variation in individuals of a species for which a reference genome is available. Conventional resequencing pipelines rely on PCR amplification of each region of interest, followed by Sanger sequencing. Regions of interest might be sets of exons, full genes, or larger intervals. The resequencing can also be done on genome BACs or fosmids clones. In a recent study [79] of haplotype resolving for more than $> 500,000$ fosmids clones, the set of clones were partitioned into 115 different groups (pools) for resequencing, using barcoding.

Our work is focused on BAC re-sequencing experiments, but it can also be applied to exome sequencing experiments or fomisd resequencing. e.g. our algorithm can optimize the pooling of the fosmids so that it reduces the overall ambiguity in read mapping for downstream analysis.

The Illumina sequencing experiments were run on a flow cell with eight lanes, each yielding in the order of 10^8bps of sequence per run, at the time of our study. A typical BAC we consider has a length ranging between 150Kbps to 250Kbps. Thus, if we sequence one BAC on each lane, a single run would produce about a 1000-fold coverage per BAC, which is far beyond necessary in a re-sequencing study. Therefore, in order to maximize the throughput of the Illumina technology, hence minimize its cost, it is of key importance to utilize each lane in a more sensible way, such as by sequencing more than one BAC per lane. However, sequencing multiple BACs per lane introduces major difficulties due to repeat sequences that are present in two or more BACs, as they would cause *tanglement* in their assemblies or ambiguous multiple mapping. In order to minimize the repeat elements that are present in multiple BACs, novel algorithmic techniques must be developed.

Available algorithms for (short reads) DNA fragment assembly such as [27, 28, 153] all suffer from the presence of repeats within the genome region to be assembled [96, 8, 146]. However, the high potential of high-throughput short-read technologies have promoted the development of novel protocols and algorithms such as the SHort Read Assembly Protocol (SHARP [140]) that aim to address the shortcomings of short read technologies. Our goal in this chapter is to help the available

fragment assembly methods or variation discovery methods using mapping strategies by providing (near) optimal utilization of the multiple lanes (or barcoding technologies) available by the Illumina technology, while keeping the cost at a minimum. For this purpose, we present algorithms that partition the input segments (e.g. BACs, fosmids or captured exons) into multiple lanes in a way to minimize potential errors due to repeat sequences shared among segments in each lane.

In what follows, we will define the pooling problem formally and will show our algorithms are quite efficient in practice and provide significant improvement to the cost of fragment assembly over random partitioning strategies.

2.1 Problem definition and general algorithmic approach

Given a set of genome fragments from known specific genomic regions of interest (e.g. a set of BACs), our ultimate goal is to construct the sequence of each fragment using the results of optimally designed Illumina sequencing experiments. Consider a set of m BACs sequenced with a read length of k (typically 25 to 150bps); the problem we address in this work is, how can we partition this set into n groups (or pools) of approximately $h = m/n$ segments, such that the identities of individual reads can be as correctly as possible attributed to the segment they come from. In other words, how can we minimize number of shared k-mers by multiple BACs in each pool in the overall configuration?

Strictly speaking, the problem does not have an exact solution, because before conducting any sequencing experiment, it is not possible to know how many shared k-mers the BACs in question would have. However, note that our focus is on re-sequencing studies, and if we have even a crude idea on the genomic coordinates of the BACs, we can approximate that missing information by using the reference sequence.

One other hurdle in designing a globally optimal experiment is the rapid proliferation of number of possible configurations. For instance, if we would like to pool $m = 150$ BACs into groups of $h = 10$, we would need to consider

$$\frac{\Pi_{i=0}^{14}\binom{150-10i}{10}}{15!} > 10^{152}$$

configurations in an exhaustive search. Since it is not feasible to search all these configurations for finding the global optimum, we propose an algorithmic approach to guide us to an approximately optimal setup.

Note that our goal is to partition a given set of m BACs into pools of size h each, with the purpose of minimizing the number of potential resequencing errors due to sequences that repeat in multiple segments within a pool. Potential assembly errors due to sequences that repeat within a single BAC, on the other hand, are beyond the scope of this study and are not investigated here. We define our problem more formally as follows.

2.1.1 The pooling problem

Given a set of m BACs that are to be placed into n pools of approximately $h = m/n$ BACs in each, let $C_{i,b}$ be the number of BACs in a pool P_i that share a particular k-mer b. If we denote the cost of b as $f(C_{i,b})$, we can write an overall cost function for a given configuration as

$$J = \sum_{i=1}^{n} \sum_{\forall b \in P_i} f(C_{i,b}) \tag{2.1}$$

and the problem becomes one of selecting the optimum partitioning $P^* = \{P_i^*\}$ that minimizes the cost J.

One can attribute alternative costs for shared k-mers b, two of which are

1. $f(C_{i,b}) = \binom{C_{i,b}}{2}$;
2. $f(C_{i,b}) = C_{i,b} - 1$.

For the reminder of this chapter, we will restrict our attention to these two formulations for reasons explained below.

The pooling problem under the cost function $f(C_{i,b}) = \binom{C_{i,b}}{2}$ is a minimization problem for the number of k-mers that are shared between pairs of BACs which are in the same pool. This can be reduced to a well known combinatorial problem called, *n-clustering problem*, as follows: construct a complete graph G where each BAC B is represented with a unique vertex v_B and given any pair of BAC B and B', set the weight of the edge (V_B, V_B') to the number of common k-mers in B and B'. The *n-clustering problem* is a problem of partitioning G into vertex sets, such that the sum of edge weights between vertices that belong to the same partition is minimized.

Unfortunately even obtaining a constant factor approximation to the n-clustering problem is NP-hard [127]. Thus in Section 2.2.1, we first reduce this problem to another combinatorial problem known as the *max n-cut*. Although this problem is also NP-hard [127], we solve it by the use of a simple local search procedure within an approximation factor of $1 - 1/n$. For $n = 15$, this implies

an approximation factor of 0.93, while for $n = 50$ (using barcoding technologies) this implies an approximation factor of 0.98. Although this approximate solution to the *max n-cut* problem does not provide a guarantee on the approximation for the pooling problem, it gives good results in practice.

An extension to the pooling problem is the *balanced pooling* problem, where we seek to minimize the cost of partitioning BACs of regions of interest into pools, such that the number of BACs in each pool is *exactly* $h = m/n$. As can be expected, even approximating the balanced pooling problem within a constant factor is NP-hard. Thus in section 2.2.2 we reduce the balanced pooling problem to the *max n-section* problem, which is the balanced version of the *max n-cut* problem. We again describe an algorithm to approximately solve this problem within a factor of $1 - 1/n$. Although the latter algorithm does not provide a guarantee on the approximation factor it obtains for the balanced pooling problem, it yields good results in practice once again.

The pooling problem under the second cost function $f(C_{i,b}) = C_{i,b} - 1$ is a minimization problem for the number of genome BACs within a pool that share each k-mer, summed over all k-mers. This is a generalized version of the pooling problem with the first cost function as will be explained below.

The pooling problem under the second cost function can be reduced to a *hypergraph partitioning problem* as follows. Let G be a hypergraph where each genome BAC B is represented with a unique vertex v_B and each subset of at most d vertices S are connected with a hyperedge e_S. In other words, d is the maximum number of vertices that can be incident to a hyperedge. In the most general case of the problem $d = m$. The weight of e_S, namely $w(e_S)$ is the number of k-mers that occur in *all* BACs in S and occur in no other BACs. Consider a partition of G into non-overlapping vertex sets. For a given subset S of vertices, let $\#(S)$ be the number of pools that have at least one vertex of S. Then the cost of e_S with respect to this partitioning is $w(e_S) \cdot (|S| - \#(S))$; here $|S|$ denotes the number of BACs in set S.

Our hypergraph partitioning problem defines a search for partitioning G into vertex sets/pools so as to minimize the total cost of the hyperedges with respect to this partition.

Unfortunately the above hypergraph partitioning problem requires $O(\binom{m}{n})$ space to just represent all the hyperedges. As this represents faster than exponential growth with the number of BACs, even setting up an instance of the problem on a computer is not feasible for the parameter values we are interested in.

Notice that if we restrict the maximum number of vertices that can be incident to a hyperedge, d, to 2 (rather than m) then our hypergraph partition problem reduces to the *n-clustering problem* and thus to the pooling problem with the first cost function. Now we can consider versions of the

hypergraph partition problem with $d = 3, 4, \cdots$ as "approximations" to our general hypergraph partitioning problem with $d = m$.

Our hypergraph partitioning problem with $d > 1$ is NP-hard [127] even to approximate within a constant factor. We reduce it to another hypergraph partitioning problem in which the cost of a hyperedge e_S with respect to a partitioning is $w(e_S) \cdot (\#(S) - 1)$ and the goal is to *maximize* the total cost of all hyperedges. In this chapter we show how to solve this problem approximately within a factor of $1 - d/2n$. For $d = 3$ and $n = 15$, our algorithm provides a 0.9 approximation factor, again sufficiently close to 1. The algorithm employs a greedy approach and is quite efficient.

We also consider a balanced version of this hypergraph partitioning problem which asks for maximizing the total cost of all hyperedges with respect to a partition, provided that the number of vertices per each pool is exactly $h = m/n$. Again we provide a $(1-d/2n)$-approximation algorithm to this problem. This algorithm is quite involved, as further described in the next section, employing a solution to the minimum weighted bipartite matching towards a greedy selection of the vertices in each partition.

2.2 Methods

In this section we give detailed descriptions of the approximation algorithms we use for solving the pooling problem, both balanced and unbalanced versions, under the two cost functions we presented earlier.

2.2.1 The pooling problem under $f(C_{i,b}) = \binom{C_{i,b}}{2}$

The pooling problem (unbalanced version) under our first cost function can be formulated as the well known max n-cut problem as follows.

Input: A weighted undirected graph $G(V, w)$, with the vertex set V representing the set $\mathcal{B} = \{B_1, B_2, \cdots, B_m\}$ of BACs, and the edge weights w. For any vertex pair v_B, v'_B, $w(v_B, v'_B)$ is the number of common k-mers in the corresponding BACs, B and B'.

Output: A partitioning of V into pools $P = \{P_1, P_2, \cdots, P_n\}$, $\bigcup_{i=1}^{n} P_i = V$, which maximizes the following objective function:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{v_B \in P_i, v'_B \in P_j} w(v_B, v'_B).$$

A local search algorithm for max n-cut (LSMnC)

1. Randomly partition the vertex set V of the graph G into n different pools.
2. If there exists a vertex $v \in V$ such that it is assigned to pool P_i ($v \in P_i$) and there exists a pool P_j such that $\sum_{u \in P_i} w(v,u) \geq \sum_{x \in P_j} w(v,x)$ then move vertex v from the pool P_i into the pool P_j.
3. Repeat second step until no change can occur.

The above simple local search algorithm, when applied to the general max n-cut problem may take too much time before it terminates. However, for our specific problem, the running time of the above algorithm is guaranteed to be polynomial with m and the maximum length of a BAC as shown below.

Proof. Let t be the total weight of the edges of the graph G, which is polynomial with m and the maximum number of k-mers in a BAC. It is clear that in each step of the local search algorithm, the total weight of edges going between pools increases by at least one. Therefore in the worst case, this algorithm terminates after t steps. □

In practice, the running time of the above local search algorithm is in the order of a second for $m = 150$, $n = 15$ and maximum BAC length of $250K$. The approximation factor achieved by the above algorithm is $1 - 1/n$ as shown below.

Proof. Consider an arbitrary vertex $v \in V$ and assume P_i is the cluster containing v after the termination of the local search. We have:

$$\forall 1 \leq j \leq n : \sum_{\forall u \in P_i} w(v,u) \leq \sum_{\forall x \in P_j} w(v,x) \Rightarrow \tag{2.2}$$

$$\sum_{\forall u \in P_i} w(v,u) \leq \frac{1}{n} \sum_{j=1}^{n} \sum_{\forall x \in P_j} w(v,x) \Rightarrow \tag{2.3}$$

$$\sum_{j=1, j \neq i}^{n} \sum_{\forall u \in P_j} w(v,u) \geq \frac{n-1}{n} \sum_{j=1}^{n} \sum_{\forall x \in P_j} w(v,x) \tag{2.4}$$

The expression $\sum_{j=1,j\neq i}^{n} \sum_{\forall u \in P_j} w(u,v)$ in left hand side of last equation represents the total weight of all edges in the "cut" incident to vertex v. Also, the expression $\sum_{j=1}^{n} \sum_{\forall x \in P_j} w(v,x)$ in the right hand side of the same equation represents the total weight of all edges incident to vertex v. Since v has been chosen arbitrary from the vertex set V, we have:

$$\sum_{\forall v \in V} \sum_{j=1,j\neq i}^{n} \sum_{\forall u \in P_j} w(v,u) \geq \sum_{\forall v \in V} \frac{n-1}{n} \sum_{j=1}^{n} \sum_{\forall x \in P_j} w(v,x) \tag{2.5}$$

According to the above inequality, the total weight of the edges which are incident to a pair of vertices that do not belong to the same pool is at least $\frac{n-1}{n}$ times the total weight of the edges in G, and thus the local search provides a $1 - 1/n$ approximation. □

A randomized $1 - 1/n$ approximation Note that a random partition will provide the same theoretical guarantee. If we place each BAC into the pools uniformly at random (i.e. in each of the n pools with probability of $1/n$), each edge will have endpoints in two different pools with the probability $1 - 1/n$. Thus, due to linearity of expectation, the expected total weight of the edges which are incident to a pair of vertices that do not belong to the same pool will be $(1 - \frac{1}{n})$ times the total weight of the edges. We implemented a simple randomized algorithm $ranPool$ and compared the results of the proposed local search algorithm with $ranPool$. The main motivation of using the local search algorithm is not only that the local search always guarantees the $1 - \frac{1}{n}$, but also the fact that in practice, the local search gives close to optimal results. We tested $ranPool$ several thousands times, and compared the best result with our algorithm. Note that, when tested a sufficient number of times, $ranPool$ indeed achieves the $1 - \frac{1}{n}$ factor, but our algorithm is close to optimal in practice. Please see the Result and Discussion section for more details and comparative figures.

remark The $ranPool$ algorithm can be easily derandmized using either pairwise independent hashing or the method of conditional probabilities. The main idea behind randomization using pairwise independent hashing in the fact that it is not really neccesary to assume mutually independence random decisions for every vertex. Rather it is only sufficient that for every pair of vertices, the probability of them making different decisions is 1/2. It can be proved that this property can be achieved using only $O(\log n)$ independent random coin tosses, and thus we one can derandomize this random process in polynomial time. Another classic way of derandomizing $ranPool$ is using the method of conditional probabilities. It can be shown that a greedy algorithm, in which BACs are

placed sequentially (i.e. one BAC at each step of the algorithm) can satisfy the necessary conditions and give the $1 - 1/n$ worst-case factor. At each step of the greedy algorithm, the conditional expectation of the total weight of the final result will be the total weight of the edges whose endpoints were places in different pools so far plus $1 - \frac{1}{n}$ times the total weight of the edges with at least one not-assigned endpoint. A simple algorithm, in which at each step a new vertex v is placed to the pool where the total weight of the edges incident to v and inside the pool is minimized among all n pools, will maximize the resulting value of the conditional expectation. Note that we did not implement this deterministic version of the algorithm as we expect to see similar results compared to our method.

2.2.2 The balanced pooling problem under $f(C_{i,b}) = \binom{C_{i,b}}{2}$

The balanced pooling problem asks to partition m BACs into n pools so as to minimize the above cost function, with the additional constraint that the number of BACs per each pool is exactly $h = m/n$. This is known as max n-section problem for which a local search algorithm by [40] guaranties an approximation factor of $1 - 1/n$.

For each vertex $u \in V$ and for each set of vertices belonging to a pool P_i, let $w(u, P_i) = \sum_{v \in P_i} w(u, v)$. The local search algorithm for the max n-section problem works as follows.

Local search algorithm for max n-section (LSMnS)

1. Initialization. Partition the vertices V into n pools $P_1, P_2, \cdots, P_n$ uniformly at random such that $|P_1| \leq |P_2| \leq \cdots \leq |P_n| \leq |P_1| + 1$.

2. Iterative step. Find a pair of vertices $v \in P_i$ and $u \in P_j$ ($i \neq j$), such that $w(v, P_i - v) + w(u, P_j - u) \geq w(v, P_j - u) + w(u, P_i - v)$ If such a pair exists move u to the cluster P_i and v to the cluster P_j.

3. Termination. If no pair of vertices is found in then terminate.

This algorithm is an extension to the local search algorithm described in section 2.2.1 and it is easy to show that it terminates in time polynomial with m and the maximum length of a BAC. Furthermore, it was shown in [40] that this algorithm has a guaranteed approximation factor of $1 - 1/n$.

2.2.3 The pooling problem under $f(C_{i,b}) = C_{i,b} - 1$

We now focus on the cost function $f(C_{i,b}) = C_{i,b} - 1$. As we discussed in the problem definition (see section 2.1), the pooling problem under cost function $f(C_{i,b}) = C_{i,b} - 1$ can be reduced to a *hypergraph partitioning problem* as follows. Let G be a hypergraph where each BAC B is represented with a unique vertex v_B and each subset S of at most m vertices, are connected with a hyperedge e_S. The weight of e_S, namely $w(e_S)$ is the number of k-mers that occur in *all* BACs in S and occur in no other BACs.

Consider a partitioning of V, the vertex set of G, into non-overlapping pools $P = \{P_1, \ldots, P_n\}$. For a given subset S of vertices, let $\#(S)$ be the number of pools of P_i that have at least one vertex of S. Then the cost of e_S with respect to P is $w(e_S) \cdot (|S| - \#(S))$ and the goal of the hypergraph partitioning problem is to *minimize* the total cost of all hyperedges.

The "dual" of this hypergraph partitioning would be another partitioning problem where e_S with respect to P is $w(e_S) \cdot (\#(S) - 1)$, and the goal is to *maximize* the total cost of all hyperedges.[1]

Input: A weighted hypergraph $G(V, w)$, with vertex set V, and weights $w(e_S)$ for each hyperedge e_S (which connects the set $S \subseteq V$ for $|S| \leq d$).

Output: A partitioning of vertices V into pools $P = \{P_1, P_2, \cdots, P_n\}$, $\bigcup_{i=1}^{n} P_i = V$, which maximizes the following objective function:

$$\sum_{S \subseteq V, |S| \leq d} w(e_S) \cdot (\#(S) - 1).$$

We give a greedy algorithm to solve the above hypergraph partitioning problem. The algorithm, at each iteration x randomly picks a vertex $v_x \in V'_x$, where V'_x is the set of vertices not processed so far, adds it to one of the pools P_i.

Before we describe the algorithm we give some definitions. Let $P_{x,i}$ be the set of vertices in pool P_i before iteration x and let $P_x = \{P_{x,1}, \ldots, P_{x,n}\}$. (Thus $V'_x = V - \bigcup_{i=1}^{i=n} P_{x,i}$.)

Given some $S \subset V$, let $\#_{P_x}(S)$ denote the number of pools $P_{x,i} \in P_x$ which include at least one vertex of S.

Also let $\widetilde{\#}_{P_{x,k}}(S)$ be a boolean function such that $\widetilde{\#}_{P_{x,k}}(S) = 1$ if $P_{x,k} \cap S = \emptyset$ and $\widetilde{\#}_{P_{x,k}}(S) = 0$ otherwise.

[1] The two problems are equivalent as $|S|$ is a constant.

A greedy algorithm for hypergraph partitioning problem (GAHP)

1. As an initial step, set $V'_0 = V$ and $\forall_{i=1}^{n} P_{0,i} = \emptyset$.

2. In each iteration $x \in \{1, \ldots, m\}$, randomly pick a vertex $v_x \in V'_x$ and put v_x into the pool

$$k_x = \arg\max_k \sum_{S \subseteq V, v_x \in S} w(e_S) \cdot \widetilde{\#}_{P_{x,k}}(S).$$

(Thus, $\forall i \leq n, i \neq k_x, P_{x+1,i} = P_{x,i}$ and $P_{x+1,k_x} = P_{x,k_x} \cup \{v_x\}$).

The above algorithm achieves an approximation factor of $1 - d/2n$ as shown below.

Proof. First we need to find a lower bound on the total cost $w(e_S) \cdot (\#(S) - 1)$ with respect to the pool set $P_{m+1} = \{P_{m+1,1}, P_{m+1,2}, \cdots P_{m+1,n}\}$ returned by the algorithm at the end of iteration m.

It is not hard to see that for any set of vertices S (for the remainder of the proof, all sets S we consider will satisfy $|S| \leq d$), for which $v_x \in S$:

$$w(e_S) \cdot \#_{P_{x+1}}(S) = w(e_S) \cdot \#_{P_x}(S) + w(e_S) \cdot \widetilde{\#}_{P_{x,k_x}}(S) \tag{2.6}$$

Thus,

$$w(e_S) \cdot \#_{P_{m+1}}(S) = \sum_{x=1, v_x \in S}^{m} w(e_S) \cdot \widetilde{\#}_{P_{x,k_x}}(S).$$

Now taking the sum of above equation for all possible hyperedges we would get:

$$\begin{aligned} \sum_{S \subseteq V} w(e_S) \cdot \#_{P_{m+1}}(S) &= \sum_{S \subseteq V} \sum_{x=1, v_x \in S}^{m} w(e_S) \cdot \widetilde{\#}_{P_{x,k_x}}(S) \\ &= \sum_{x=1}^{m} \sum_{S \subseteq V, v_x \in S} w(e_S) \cdot \widetilde{\#}_{P_{x,k_x}}(S) \end{aligned} \tag{2.7}$$

For bounding the left hand side of equation 2.7, we will consider an arbitrary iteration x.

$$n \cdot \sum_{S \subseteq V, v_x \in S} w(e_S) \cdot \widetilde{\#}_{P_{x,k_x}}(S) \geq \sum_{S \subseteq V, v_x \in S} \sum_{i=1}^{n} w(e_S) \cdot \widetilde{\#}_{P_{x,i}}(S). \tag{2.8}$$

By adding up the right hand side of equation 2.8 over all values of x we get:

$$\sum_{x=1}^{m}\sum_{S\subseteq V, v_x\in S}\sum_{i=1}^{n} w(e_S)\cdot \widetilde{\#}_{P_{x,i}}(S) = \sum_{S\subseteq V}\sum_{x=1, v_x\in S}^{m}\sum_{i=1}^{n} w(e_S)\cdot \widetilde{\#}_{P_{x,i}}(S). \tag{2.9}$$

.

For bounding equation 2.9 we first have to argue for any arbitrary $S \subseteq V$ we have,

$$\sum_{x=1, v_x\in S}^{m}\sum_{i=1}^{n} w(e_S)\cdot \widetilde{\#}_{P_{x,i}}(S) \geq w(e_S)\cdot(n+\cdots(n-|S|+1))$$

Thus,

$$\sum_{S\subseteq V}\sum_{x=1, v_x\in S}^{m}\sum_{i=1}^{n} w(e_S)\cdot \widetilde{\#}_{P_{x,i}}(S) \geq \sum_{S\subseteq V} w(e_S)\cdot(n+\cdots(n-|S|+1)) \tag{2.10}$$

Now using equations 2.8,2.9 and 2.10 we will have:

$$\sum_{x=1}^{m}\sum_{S\subseteq V, v_x\in S} w(e_S)\cdot \widetilde{\#}_{P_{x,1},k_x}(S) \geq \frac{\sum_{S\subseteq V} w(e_S)\cdot(|S|\cdot n-(1+2+\cdots(|S|-1)))}{n}. \tag{2.11}$$

Utilizing equation 2.11 and 2.7 we conclude:

$$\begin{aligned}\sum_{S\subseteq V} w(e_S)\cdot(\#_{P_{m+1}}(S)-1) \geq \\ \frac{\sum_{S\subseteq V} w(e_S)\cdot(|S|\cdot n-\frac{|S|(|S|-1)}{2})}{n} - \sum_{S\subseteq V} w(e_S) \\ = \frac{n\cdot\sum_{S\subseteq V} w(e_S)\cdot(|S|-1)-\sum_{S\subseteq V} w(e_S)\cdot\frac{|S|(|S|-1)}{2}}{n}\end{aligned} \tag{2.12}$$

Now to find the approximation factor for this greedy algorithm we need to find an upper bound of the optimal solution. It is easy to see that for even optimal partitioning $\sum_{e_S} w(e_S)\cdot(\#(S)-1) \leq$

$\sum_{e_S} w(e_S) \cdot (|S| - 1)$. Thus, the approximation factor α can be bounded as:

$$\begin{aligned}
\alpha &= \frac{\sum_{S \subseteq V} w(e_S) \cdot (\#_{P_{m+1}}(S) - 1)}{\sum_{S \subseteq V} w(e_S) \cdot (\#_{P_{opt}}(S) - 1)} \\
&\geq \frac{\sum_{S \subseteq V} w(e_S)((|S| - 1)n - \frac{|S|(|S|-1)}{2})}{n \cdot \sum_{S \subseteq V} w(e_S) \cdot (|S| - 1)} \\
&= 1 - \frac{\sum_{S \subseteq V} w(e_S) \cdot \frac{|S|(|S|-1)}{2}}{n \cdot \sum_{S \subseteq V} w(e_S) \cdot (|S| - 1)} \\
&= 1 - \frac{\sum_{S \subseteq V} w(e_S) \cdot (|S| \cdot (|S| - 1))}{2n \cdot \sum_{S \subseteq V} w(e_S) \cdot (|S| - 1)}
\end{aligned} \tag{2.13}$$

Now, as $|S| < d$ we can easily see that $\alpha \geq 1 - d/2n$. □ □

2.2.4 The balanced pooling problem under $f(C_{i,b}) = C_{i,b} - 1$

Our last algorithm deals with the *balanced* pooling problem under the cost function $f(C_{i,b}) = C_{i,b} - 1$, for which we give a greedy approximation algorithm. We remind the reader that there are $m = nh$ vertices to be assigned into n pools, and eventually each pool must have exactly h vertices.

A greedy algorithm for balanced hypergraph partitioning (GABHP)

The algorithm starts with a set of n empty pools, $P = \{P_1, \cdots, P_n\}$, and at each iteration x, it selects a set of n arbitrary vertices, say $Y_x = \{y_{1,x}, \cdots, y_{n,x}\}$, which are not assigned to any of the pools yet, and adds them to the pools such that each pool receives *exactly* one new vertex. Let the set of vertices in pool P_i at the beginning of iteration x be denoted by $P_{i,x}$ and let $P_x = \{P_{x,i}, \ldots, P_{x,n}\}$. Thus, in iteration x, each $y_{j,x}$ is assigned to exactly one of the pools $P_{x,i}$.

For any set of vertices $S \in V$, let $\lambda(y_{j,x}, P_{x,i}, S)$ be a boolean function defined as follows.

$$\lambda(y_{j,x}, P_{x,i}, S) = \begin{cases} 1 & \text{if } \exists\, y_{\ell,x} \neq y_{j,x} : y_{\ell,x} \in S,\ \ y_{\ell,x} \in P_{x,i} \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, for a given vertex $y_{j,x}$, a pool $P_{x,i}$, and a vertex set S, $\lambda(y_{j,x}, P_{x,i}, S)$ is equal to zero if and only if no other vertex $y_{\ell,x}$ incident to the hyperedge e_S has already been assigned to the pool $P_{x,i}$.

Then, for each pool $P_{x,i}$, we define the *marginal* cost function $\mu(y_{j,x}, P_{x,i})$ with respect to the potential assignment of $y_{j,x}$ to $P_{x,i}$, as follows.

$$\mu(y_{j,x}, P_{x,i}) = \sum_{S \subseteq V, y_{j,x} \in S} w(e_S) \cdot \lambda(y_{j,x}, P_{x,i}, S)$$

We now construct a *new* complete bipartite graph H with vertex sets Y_x and P_x such that for any vertex $y_{j,x} \in Y_x$ and pool $P_{x,i} \in P_x$, there exists an edge in H with weight $\mu(y_{j,x}, P_{x,i})$. Then we find a perfect *minimum* weighted matching for H, i.e. a perfect matching where the sum of weights of the edges in the matching has the minimum possible value by using the well known Hungarian algorithm ([109]).[2]

For each j, $1 \leq j \leq n$, let $\pi(y_{j,x})$ be the pool which is matched to $y_{j,x}$ in the prefect minimum bipartite matching of the graph H. We add the vertex $y_{j,x}$ to the pool $\pi_{x,i}$.

We run the above iterative step for $x = 1 \ldots h$ so as to assign each one of the $m = nh$ vertices into one pool in a balanced manner.

The above algorithm gives an approximation to the balanced hypergraph partitioning problem within a factor of $1 - \frac{d}{2n}$, where d is the maximum number of vertices that can be incident to a hyperedge. The proof for the approximation factor is similar to that for the unbalanced hypergraph partitioning problem and thus is omitted.

2.3 Results and discussion

We report results on two sets of BACs (with $m = 150$) on which we tested our algorithms for both balanced and unbalanced pooling problem using both cost functions. We start by noting that for both data sets the results obtained by the balanced pooling algorithms turned out to be almost identical to those obtained by the unbalanced pooling algorithms for each of the two cost functions we used. In other words, the cost of the partition obtained by the LSMnC algorithm was almost identical to that of the LSMnS algorithm and the cost obtained by the GAHP algorithm was very similar to that of the GABHP problem. It is also interesting to note that the unbalanced pooling algorithms returned very balanced partitions.[3] We compare the performance of these algorithms with that of random partitioning of BACs into pools.

[2] We actually solve the dual, weight maximization problem after subtracting each edge weight from the maximum edge weight.

[3] The number of BACs obtained by the unbalanced pooling algorithms were never less than 7 and never more than 12 in any pool.

We measure the performance of our algorithms and that of random partitioning with respect to our general cost function $f(c_{i,b}) = C_{i,b} - 1$. The total cost of a particular partitioning of a set of m BACs into pools $P_1, \ldots, P_n$, is $J = \sum_{i=1}^{n} \sum_{\forall b \in P_i} f(C_{i,b})$. In order to compute the cost J for a partitioning, we construct, for each pool P_i, the *joint trie* of the k-mers (for this study $k = 50$) of all BACs in P_i, denoted T_i. The trie T_i can be constructed in time linear with the total lengths of the BACs in P_i as per the linear time algorithms for suffix tree construction [100, 126]. During the construction of T_i, at each leaf corresponding to a k-mer b, we maintain the labels of the specific BACs that include b. By going through all leaves of each T_i, we compute J in time linear with the total lengths of the BACs.

We used two data sets in our experiments, each consisting of 150 BACs. The first set of clones were collected in the high-resolution analysis of *lymphoma genomes project* at the BC Genome Sciences Center. They represent regions of interest in the genome of a tumor sample, where there are marked local deviations from the reference human genome. The BAC coordinates are deduced by aligning BAC fingerprints to the reference genome [82] and are confirmed by BAC end sequencing experiments.

The second set is a synthetic library of clones with a mean size of 182kb and a standard deviation of 34kb, representing a random sampling of the finished regions of the reference human genome, hg18.

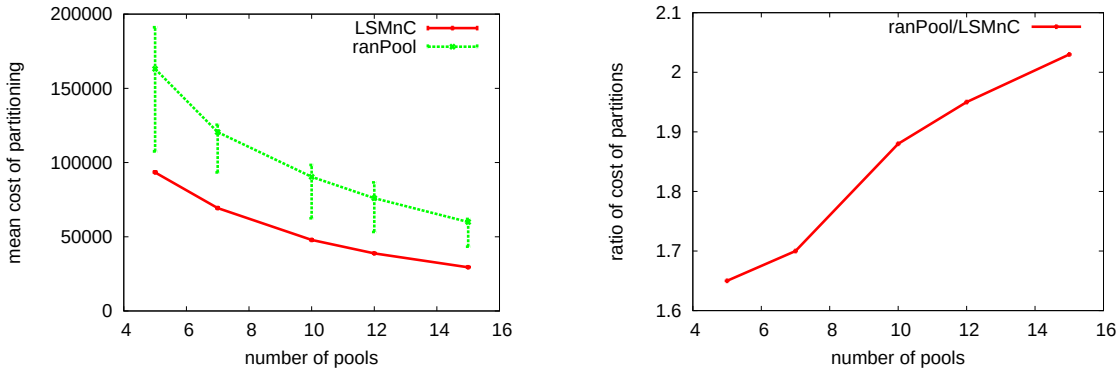
2.3.1 Pooling experiments with LSMnC/LSMnS algorithms

We first compare the performance of our local search methods LSMnC/LSMnS with random partitions (denoted *ranPool*). Although these methods were designed to minimize the cost of pooling with respect to the cost function $f(C_{i,b}) = \binom{C_{i,b}}{2}$, we report their performance with respect to the second cost function, $f(C_{i,b}) = C_{i,b} - 1$, as it better captures the notion of performance we would like to measure.

Note that ranPool is known to give an *expected* approximation factor of $1 - 1/n$ for the max n-cut problem. However, LSMnC will guarantee a *worst case* approximation factor of $1 - 1/n$ for the max n-cut problem.

In figures 2.1(a) and 2.2(a) we give the distribution (the mean value as well as the highest and lowest 25%) of the cost obtained by 5000 independent runs of LSMnC/LSMnS and ranPool methods on the *lymphoma* and the *synthetic* data sets. The figures show how the cost changes with respect to the increasing number of pools. We also give how the ratio between the cost of the ranPool and

the LSMnC/LSMnS methods change with respect to the number of pools (again showing the mean value, the highest and the lowest 25% of the ratio of the costs of ranPool and the LSMnC/LSMnS methods) in figures 2.1(b) and 2.2(b). It is easy to see that by increasing the number of pools the cost of ranPool and LSMnC/LSMnS would reduce. However, more interestingly by increasing the number of pools, the ratio between cost of ranPool and LSMnC/LSMnS increases(Figure 2.1(b) and 2.2(b)), meaning that our proposed methods are more effective with higher number of pools.



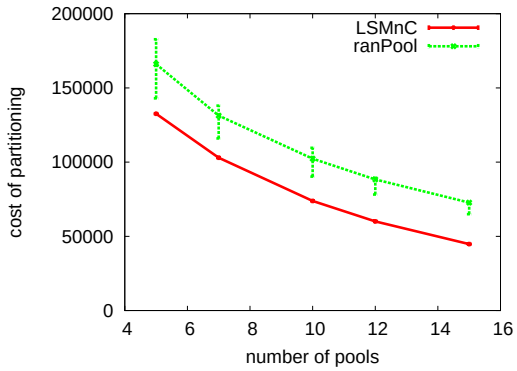
(a) The change of cost with respect to the number of pools.

(b) The ratio between the costs of random partitioning and the LSMnC/LSMnS algorithms.

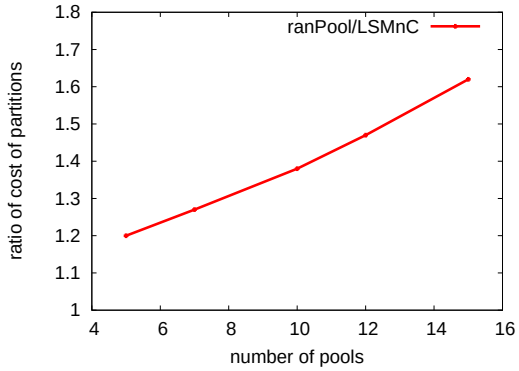
Figure 2.1: The cost of ranPool (green) and LSMnC/LSMnS (red) methods with respect to the number of pools: mean, upper quartile and lower quartile results reported on 5000 independent runs on the lymphoma data set.

We finally give the distribution of the costs obtained in the 5000 independent runs of both ranPool and the LSMnC/LSMnS methods on the lymphoma and the synthetic data sets in figures 2.3 and 2.4 respectively.

As can be observed, the results obtained by the LSMnC/LSMnS algorithms are typically much better than that obtained by ranPool. At $n = 15$ the LSMnC/LSMnS approach provides a factor 2 improvement to the (mean) cost of random partitioning for the lymphoma data set. The cost improvement is more than a factor of 1.4, even for the random partition with the lowest cost among the 5000 independent trials(for synthetic data the cost improvement factor was 1.35 in comparison to the lowest cost among 5000 independent trials).



(a) The change of cost with respect to the number of pools.



(b) The ratio between the costs of random partitioning and the LSMnC/LSMnS algorithms.

Figure 2.2: The cost of ranPool (green) and LSMnC/LSMnS (red) methods with respect to the number of pools: mean, upper quartile and lower quartile results reported on 5000 independent runs on the synthetic data set.

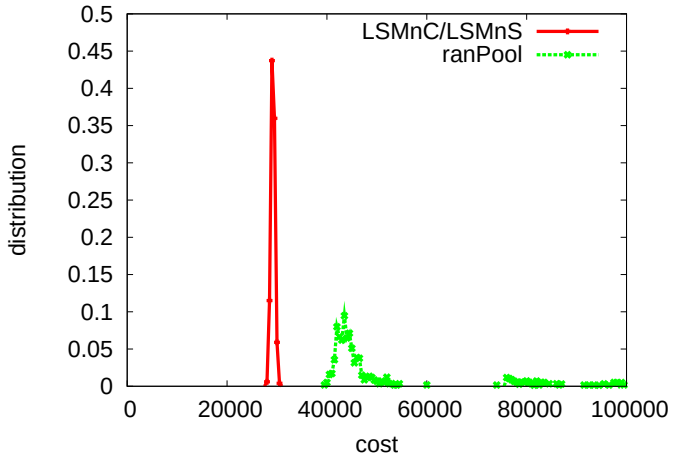


Figure 2.3: The distribution of cost obtained by ranPool and LSMnC/LSMnS for $n = 15$ on the lymphoma data set after 5000 independent runs.

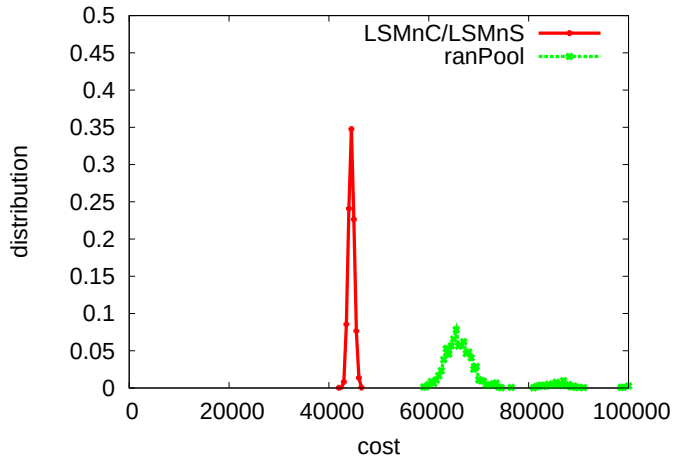


Figure 2.4: The distribution of cost obtained by ranPool and LSMnC/LSMnS for $n = 15$ on the synthetic data set after 5000 independent runs.

2.3.2 Pooling experiments with GAHP/GABHP algorithms

The local search algorithms LSMnC and LSMnS aim to "minimize" the cost of pooling with respect to the cost function $f(C_{i,b}) = \binom{C_{i,b}}{2}$; however the cost obtained by these algorithms were considerably lower than that obtained by ranPool even with respect to the second cost function - which provides a more accurate performance measure.

Our second set of algorithms, GAHP/GABHP are designed to "minimize" the cost with respect to the second cost function. They are flexible in the sense that one can set up the value of d as desired; for $d = n$, the optimal solution to the hypergraph partitioning indeed minimizes the cost function $f(C_{i,b}) = C_{i,b} - 1$. We tried the two algorithms for both $d = 2$ and 3 in order to evaluate their advantage over the local search algorithms as well as random partitioning. The running time of both GAHP and the GABHP algorithms are exponential in d (the number of hyperedges grow exponentially with increasing d) thus it is of crucial importance to know up to which value of d, an improvement in performance could be expected. A significant performance improvement by GAHP/GABHP methods using $d = 3$ over LSMnC/LSMnS methods (which solve the hypergraph partitioning problem for $d = 2$) may imply that d should be increased to 4 or more for better

performance.[4]

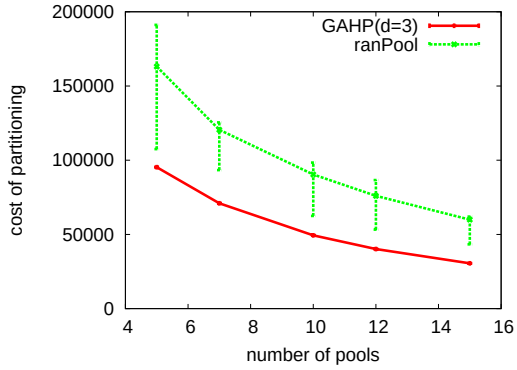
In figures 2.5(a) and 2.6(a) we compare the distribution (the mean value as well as the highest and lowest 25%) of the cost obtained by 5000 independent runs of GAHP/GABHP and ranPool methods on the *lymphoma* and the *synthetic* data sets. The figures show how the cost changes with respect to the increasing number of pools. We also show how the ratio between the cost of the ranPool and the GAHP/GABHP methods change with respect to the number of pools (again showing the mean value, the highest and the lowest 25% of the ratio of the costs of ranPool and the GAHP/GABHP methods) in figures 2.7(a) and 2.7(b).

We also investigate the effect of using hypergraphs with $d = 3$ over the use of ordinary graphs with $d = 2$ on the GAHP/GABHP methods. We again report the results of 5000 independent trials on both data sets in figures 2.7(a) and 2.7(b).

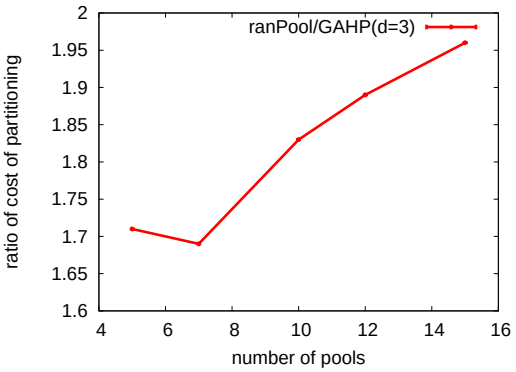
The performance improvement achieved by increasing d form 2 to 3 is negligible for both data sets. It may be possible to explain the relatively poor performance of GAHP/GABHP methods for $d = 3$ (in comparison to $d = 2$) by investigating the distribution of repeat sequences among the BACs in the two data sets. The number of k-mers which are repeated in exactly two BACs are $100 - 200$ times more than those repeated in three BACs or more; see figures 2.8(a) and 2.8(b) for the distribution of hyperedge weights in the the two data sets. Thus the total weight of hyperedges incident to three vertices or more is insignificant in comparison to edges that are incident to exactly two vertices. Thus, the hypergraph partitioning algorithms largely "ignore" the hyperedges whose contribution to the total cost is very small. We expect that the performance of the GAHP/GABHP methods for $d = 3$ is likely to be superior to that for $d = 2$ if highly repetitive BACs are sequenced.

We finally compare the two algorithmic approaches proposed in this chapter: GAHP/GABHP (for $d = 2$) and LSMnC/LSMnS. Note that the cost function of GAHP/GABHP when $d = 2$ is equivalent to the cost function used by LSMnC/LSMnS. In figure 2.9(a) and 2.9(b) a comparison of the two approaches are provided for both data sets. Interestingly enough, the performance of LSMnC/LSMnS approach is slightly better than the GAHP/GABHP approach for both data sets. This behaviour demonstrates that a LSMnC/LSMnS (which finds a local optimum) outperforms a greedy based method (GAHP/GABHP), which does not find a local optimum. However, it should be noted that we believe if regions of DNA with high repetitions are used, GAHP/GABHP (when $d < 2$) should give better results than LSMnC/LSMnS and GAHP/GABHP (when $d = 2$).

[4]Unfortunately the approximation factor achieved by the GAHP/GABHP methods deteriorate with increasing k. Note that for $d = 3$, the approximation factor guaranteed by the GAHP/GABHP approach to the hypergraph partitioning problem is $1 - 3/2n$ which is equal to 0.9 for $n = 15$.

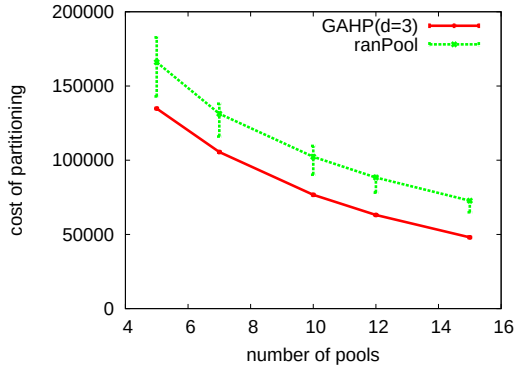


(a) The change of cost with respect to the number of pools.

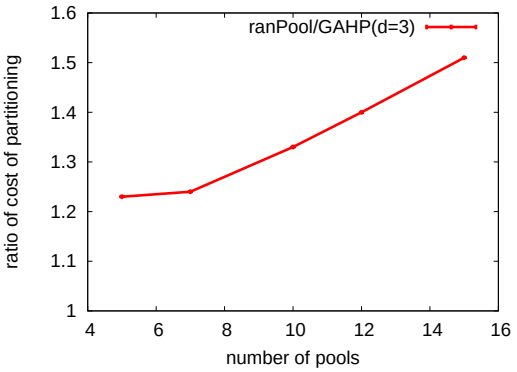


(b) The ratio between the costs of random partitioning and the GAHP/GABHP algorithms.

Figure 2.5: The cost of ranPool (green) and GAHP/GABHP (red) methods ($d = 3$) with respect to the number of pools: mean, upper quartile and lower quartile results reported on 5000 independent runs on the lymphoma data set.

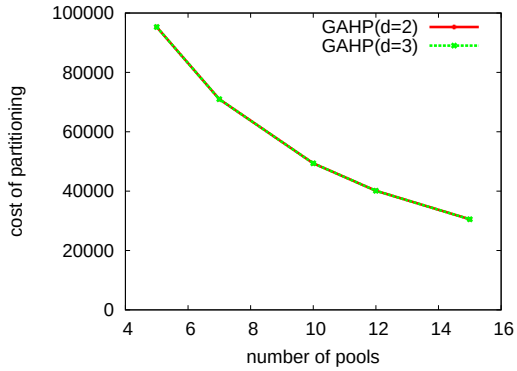


(a) The change of cost with respect to the number of pools.

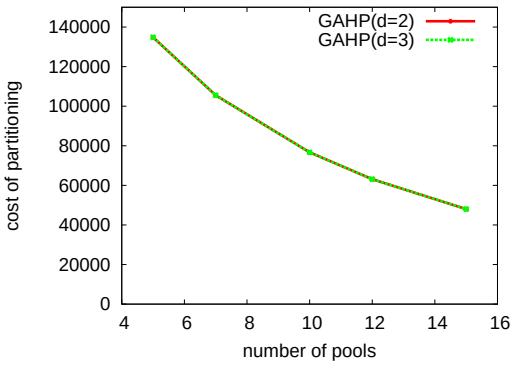


(b) The ratio between the costs of random partitioning and the GAHP/GABHP algorithms.

Figure 2.6: The cost of ranPool (green) and GAHP/GABHP (red) methods ($d = 3$) with respect to the number of pools: mean, upper quartile and lower quartile results reported on 5000 independent runs on the synthetic data set.

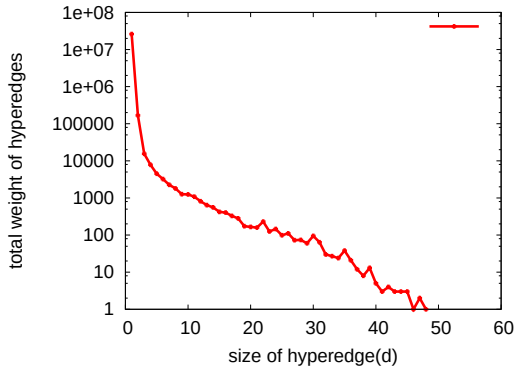


(a) Comparing the performance of GAHP/GABHP method for $d = 2$ and $d = 3$ on the lymphoma data set.

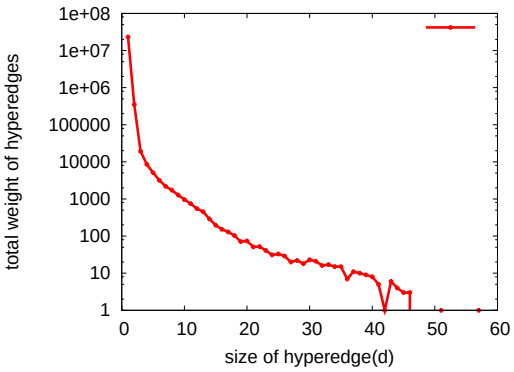


(b) Comparing the performance of GAHP/GABHP method for $d = 2$ and $d = 3$ on the synthetic data set.

Figure 2.7: the mean value and error bounds of 5000 runs of GAHP ($d = 2$ and $d = 3$).

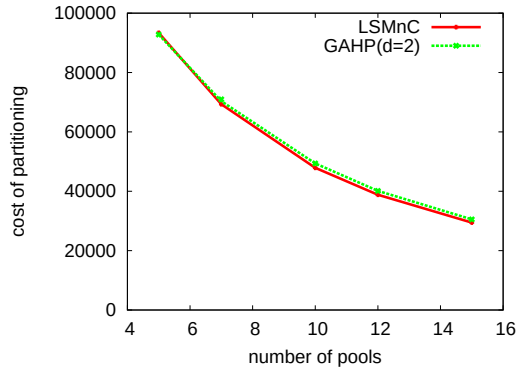


(a) Distribution of hyperedge weights on the lymphoma data set.

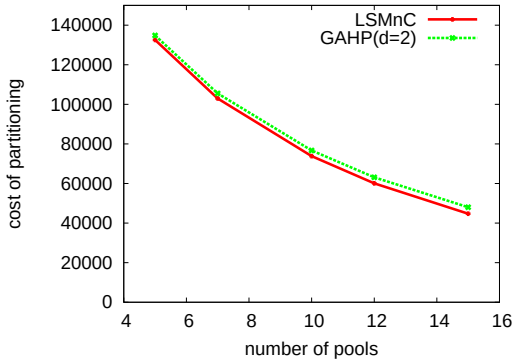


(b) Distribution of hyperedge weights on the synthetic data set.

Figure 2.8: The distribution of hyperedge weights (in log scale) among 5000 BACs in the two data sets considered.



(a) Comparing GAHP/GABHP approach to LSMnC/LSMnS approach on the lymphoma data set.



(b) Comparing GAHP/GABHP approach to LSMnC/LSMnS approach on the synthetic data set.

Figure 2.9: The cost of LSMnC/LSMnS approach (red) with GAHP/GABHP method (green) with respect to the number of pools: mean, upper quartile and lower quartile results reported on 5000 independent runs on both data sets.

Chapter 3

Novel sequence and transposon insertions discovery

As we discussed in the first chapter, the problem of discovering human genome structural variation (SV) enjoyed increased attention from the genomics research community, in the past few years. Many studies were published to characterize short insertions, deletions, duplications, and inversions, and associate copy number variants (CNVs) with disease. Although various computational methods have been developed for the types of SVs mentioned above, no good algorithm fully capable of discovering transposon insertions or novel sequence insertion were developed. The discovery of transposon insertions is particularly challanging because of the repetetive nature of the sequence contents. Note that transposon insertions form a very important class of SVs to the study of human evolution and disease. Furthermore, detection of novel sequence insertions (i.e. sequences in a doner genome which are missing in the reference genome) requires sequence data, however, the "detectable" sequence length with read-pair analysis is limited by the insert size. Thus longer sequence insertions that contribute to our genetic makeup are not extensively researched.

In this chapter, we will first present NovelSeq: a computational framework to discover the content and location of long novel sequence insertions using paired-end sequencing data generated by the next generation sequencing platforms. Our framework can be built as part of a general sequence analysis pipeline to discover multiple types of genetic variation (SNPs, structural variation, etc.), thus it requires significantly less computational resources than *de novo* sequence assembly. We apply our methods to detect novel sequence insertions in the genome of an anonymous donor, and validate our results by comparing with the insertions discovered in the same genome using various

sources of sequence data.

Next, we will provide a complete and novel formulation to discover both loci and classes of transposons inserted into genomes, sequenced with high-throughput sequencing technologies. Our focus will be especially on *Alu* insertions. We will also present how we developed an efficient algorithm to discover transposons, and how the experimental results of our algorithm compare with known results or available databases.

3.1 Novel sequence insertion discovery

Here we present a computational framework to discover the locus and content of novel sequence insertions using the NGS platforms. We test our methods with the high-coverage (42X) short-insert sequence library generated from the genome of a Yoruba African individual (NA18507) sequenced using the Illumina platform [21]. We validate the content of the predicted novel sequence insertions by comparing with sequences generated from fosmid end-sequence assembly [75], full fosmid sequencing [77], and *de novo* sequence assembly of the same Illumina WGS library [92]. We show that our methods are reliable, and together with the cost optimizations introduced by the NGS platforms, they can efficiently be used to characterize the DNA sequences missing from the reference assembly to obtain a better picture of the human genome diversity.

A ”novel sequence insertion” refers to an insertion of a sequence into the donor genome where no subsequence with high similarity to the inserted sequence exists in the reference genome. We aim at identifying novel sequence insertions in a high-coverage sequenced donor genome through our computational pipeline NovelSeq.

Note that the insertions of repeat sequences such as SINEs and LINEs do not constitute as novel sequence insertions since paralogs of the same repeat sequence exists elsewhere in the reference genome assembly. Therefore, the algorithms presented here will not be able to predict such repeat sequence insertions unless the inserted sequence is highly divergent from other existing copies. Those algorithms will be covered in the next section.

The presentation of the general approach of the NovelSeq pipeline is divided into five different phases:

3.1.1 Overview of the NovelSeq pipeline

1-Mapping of the paired-end reads onto the reference genome: The computational pipeline begins by mapping the WGS paired-end reads onto the reference genome using mrFAST [7] and

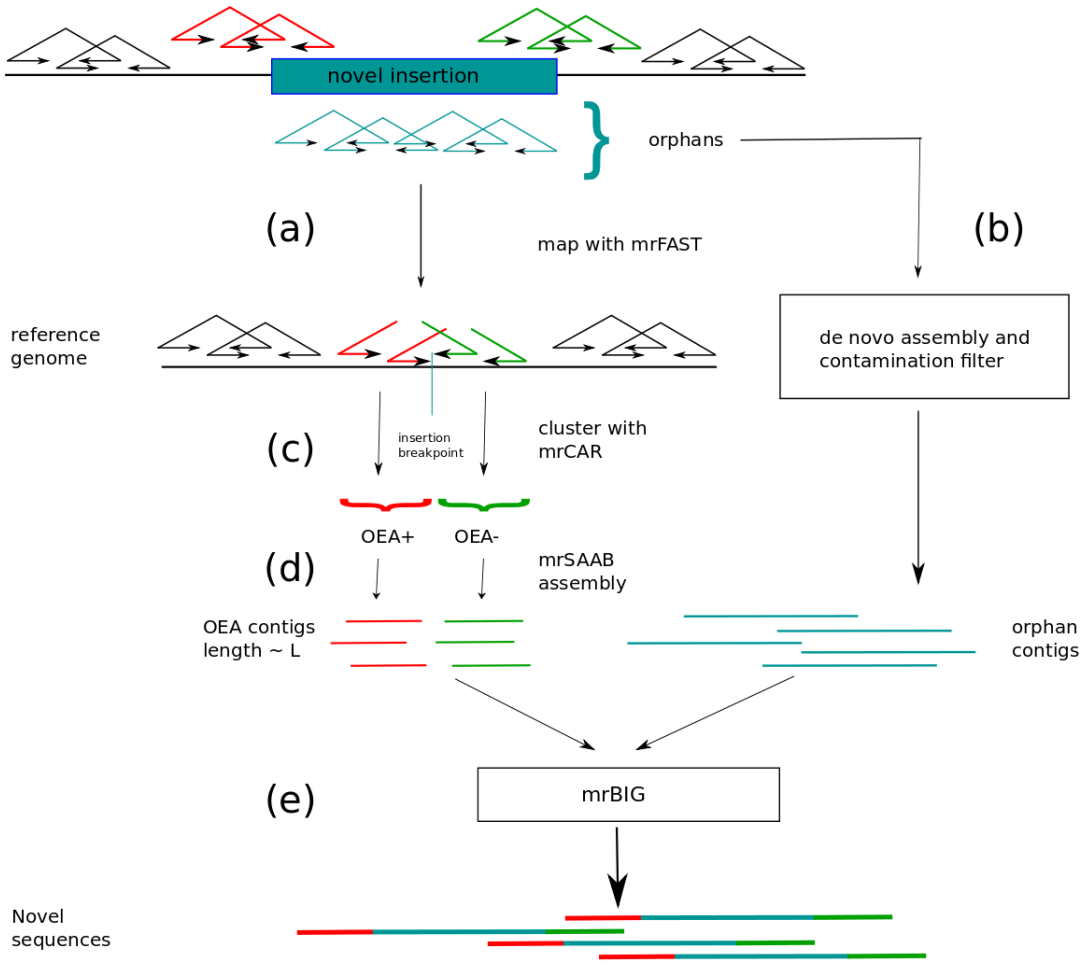


Figure 3.1: In this figure we illustrate the 5 stages of the NovelSeq pipeline. (a) Starts by mapping the paired-end reads to the reference genome, and classifies the paired-end reads to OEA and Orphan reads. (b) Assembles the orphan paired-end reads using available de novo assemblers, and removes any contigs which are result of contamination. (c) Clusters the OEA reads into groups and finds the insertion locus supported by each OEA cluster. (d) Assembles the unmapped end-read of paired-end reads in each OEA cluster (the OEA reads with different orientation of mapping should be assembled independently). (e) Merges the orphan contigs and OEA contigs to find the locus of each orphan contig insertion.

identifying *orphan* reads and *one end anchored (OEA)* reads. The paired-end reads where neither end read [1] sequences can be mapped (with more than 95% similarity) to the reference genome are classified as *orphan* reads. Following the nomenclature previously described [75], if only one end-read is mapped onto the reference genome, such paired end reads are classified as One-End-Anchored (OEA).

A hypothesis which can explain the existence of these orphan and OEA paired-end reads in a sequenced donor genome is as follows: The unmapped reads of the OEA pairs, and the orphan paired-end sequences both belong to novel sequence insertions (See phase (a) in Figure 3.1).

2-Orphan assembly and contamination removal: Using available de novo assembly algorithms such as EULER-SR [28] or ABySS [134], we assemble *all* the orphan reads into longer contigs. These contigs may later be identified as novel insertion sequences in the donor genome. In addition, we perform an initial screening of the contigs using BLAST [12], and remove any contig that contains sequences from known contaminants (e.g. Epstein-Barr, E. coli, vectors, etc). As a second test to remove the mapping artifacts, we remove the contigs that can be aligned to the reference genome with a sequence similarity more than 99%.

3-Clustering the OEA reads: We use a novel *clustering algorithm* mrCAR (micro-read Cluster Anchored Reads) to cluster the OEA reads based on their mapping orientations and locations on the reference genome such that those OEA reads which support the same insertion in the donor genome are grouped together. Note that, from a biological point of view, for each potential novel sequence insertion, there exists a group of OEA read alignments with $'+'$ orientation (i.e. the single end read that has an alignment on the reference genome is aligned to the forward strand) and a second group of OEA read alignments with $'-'$ orientation (i.e. the single end read is aligned to the reverse strand). Throughout the section, we use the term *OEA cluster* to describe the two groups of OEA reads which are both mapped to different strands yet support the same novel sequence insertion.

The goal of mrCAR is to identify these OEA clusters efficiently such that, with a minimum number of novel sequence insertion prediction, all the OEA paired-end reads are explained (i.e. for every OEA paired-end read oea_i, there exists an insertion prediction that is supported by oea_i).

4-The local assembly of the OEA clusters: All single end reads in the OEA clusters which were formed in the previous phase are assembled into two OEA contigs using a local assembly routine,

[1] Each end sequence of a paired-end read is referred to as end-read.

mrSAAB (micro-read Strand-Aware Assembly Builder).

For each OEA cluster, the goal is to assemble the unmapped $OEA+$ reads into a single contig (i.e. $OEA+$ contig) and the unmapped $OEA-$ into a single contig (i.e. OEA contig).

5-Merging the orphan contigs and the OEA contigs: In this phase of the pipeline, we aim to merge the OEA contigs (from both forward and reverse strands) with the orphan contigs. Through this merging step, we both provide more read support for the orphan contig, and obtain the approximate anchoring position of the novel sequence insertion to the reference genome.

Our merging algorithm mrBIG (micro-read Big Insertion Gluer) aims to report the maximum number of orphan contigs which can be merged with OEA contigs with a *high support*. mrBIG was developed based on a maximum weighted matching approach in a bipartite graph.

3.1.2 Methods

Notations and definitions

In what follows, we present the notations and new definitions which will be used in the rest of this section. We define the set of paired-end reads of a sequenced donor genome as $R = \{pe_1, pe_2, \cdots, pe_n\}$. Each paired-end read pe_k may have multiple mapping locations on the reference genome. The set of all alignments of pe_k is defined as $Align(pe_k) = \{a_1pe_k, a_2pe_k, \cdots, a_jpe_k\}$. Structural variation discovery algorithms using read-pair analysis start by calculating the observed distance between the two end reads of a paired-end read. This distance is referred to as the *insert size* (denoted by $InsSize$). The $InsSize$ is assumed to be in a range of $[\Delta_{\min}, \Delta_{\max}]$ and can be calculated as previously described[147, 55].

An alignment of a paired-end read to the reference genome is *concordant*, if the distance between the aligned end reads is within the expected range of $[\Delta_{\min}, \Delta_{\max}]$ and the paired-end alignment orientation is $+-$ (i.e. the end read which was aligned on the left side of its mate [i.e. that is the other read from the same paired-end read] has an alignment orientation of $'+'$ and the mate has an orientation of $'-'$).

The set of One End Anchored reads is represented as OEA and the set of orphan reads is represented as $Orph$. Note that $Orph, OEA \subset R$. The paired-end reads in OEA may also be mapped to multiple locations on the reference genome. Given $pe \in OEA$, $ape = (loc(ape), or(ape))$, where $loc(ape)$ is the location the read is aligned to the reference genome and $or(ape)$ is the alignment orientation (i.e. $or(ape) \in \{+, -\}$) since only one end read aligns to the reference genome).

Clustering the OEA reads

In this section we formally describe a greedy algorithm which we named mrCAR to identify the OEA clusters [2]. We first mathematically formulate the conditions required by a group of OEA reads so that they all support the same novel insertion. Next, similar to the approach introduced in [55] for clustering the discordant paired-end reads, we present an efficient greedy algorithm to find the minimum number of insertions such that all of the OEA reads would be "supporting" at least one insertion (i.e. a maximum parsimonious explanation of all OEA reads [55]). We remind the reader that although the mapping location of an OEA read serves as a guide to detect the locus of the novel insertion sequence, the possibility of multiple mapping locations for an OEA read makes detecting the correct locus a challenging task.

Clustering rules: A set of OEA reads $clu \subset OEA$ supports the same insertion if the following conditions hold:

- For every pair of OEA read alignments $\rho_F \in clu$ and $\rho_R \in clu$ (without loss of generality we assume that ρ_F aligns onto the forward and ρ_R aligns onto the reverse strand), the mapping location of ρ_F is before the mapping location of ρ_R.
- The *maximum* pairwise distances of the mapping locations of the OEA reads in clu with the same mapping orientation must be less that the maximum $InsSize$, $\Delta_{\max}$.
- The difference between the mapping locations of two OEA reads with different mapping orientations should not exceed twice the maximum $InsSize$ ($2\Delta_{\max}$).

Note that an OEA cluster c is called a "maximal valid cluster" if no more OEA read alignment can be added to c that all the conditions noted above remain valid. Previous studies in identifying structural variation events such as [55] and [136] also used a similar definition of a maximal valid cluster.

By using an iterative method, we find all such maximal valid clusters in polynomial time. We first order all the OEA read alignments based on their loc value, and start traversing the genome from left to right. For each position of the genome k, we consider a window of size $2\Delta_{\max}$ centered at k. Every OEA alignment inside the first half of the window with an orientation $'+'$, and every OEA alignment on the second half of the window with orientation of $'-'$, is considered as one potential

[2] An OEA cluster is a group of OEA reads all supporting the same novel insertion.

maximal valid clusters. Finally, a pairwise comparison is performed between all overlapping clusters detected in the previous step and only the maximal clusters are reported.

Algorithm to select the minimum number of clusters We define the Maximum Parsimonious Insertion Detection (MPID) problem as following: Given a set of OEA clusters where each cluster potentially indicates a novel insertion, our goal is to select the minimum number of clusters (i.e. minimizing the total number of insertions) such that all the OEA reads are aligned onto the reference genome. We model this problem as a set cover problem and provide an $O(\log n)$ approximation solution to it. Note that the set of all OEA reads is the *universe* of elements, and the clusters created in the previous step are the sets which are selected to cover this universe. MPID is a necessary step since an OEA can be present in multiple clusters.

Local assembly of the OEA clusters

The next step is to assemble the unmapped reads of OEA clusters that were created by the clustering algorithm and selected by the set cover approach. Within a cluster, the OEA reads with an alignment to the forward strand (i.e. $+$ strand) must be assembled together and those with an alignment to the reverse strand (i.e. $-$ strand) must be assembled into OEA contigs independently. However, the available *de novo* assemblers including EULER and ABySS do not provide the option of assembling the reads of only a single strand [3]. In the single-end option, both ABySS and EULER consider the reverse complements of the read sequences as well. We therefore develop a local assembly routine that makes use of the fact that all unmapped reads from a single OEA cluster originate from the single strand reciprocal to the mapping orientation of the anchored reads from the same cluster. During the traversal of the assembly graph, we do not allow two consecutive OEA reads such that the mapping locations of their mates (from the corresponding paired-end reads) are *too far* from each other. The map location of the anchored read dictates the approximate position of the unmapped read in the local OEA assembly. The confidence interval for this position information depends on the $InsSize$ distribution.

Our local assembly routine is based on the standard overlap-layout-consensus graph approach. Note that this routine can also be implemented with an Eulerian path approach assembly based on a de Bruijn graph (e.g. through a modification to ABySS). In what follows, we briefly present this routine.

[3] Personal communication with the main developers of the tools

Traversal of the overlay graph We first construct the overlay graph for all unmapped reads in an OEA cluster whose mates are anchored to the same strand.

Note that for each OEA cluster, there will be two disjoint assembly graphs representing two different strands. Given a pair of nodes u, v in the overlay graph (representing two OEA reads), we add a weighted directed edge connecting u to v if there exists a suffix of u with a prefix of v. The weight of the noted edge will be a function of the suffix-prefix overlap between them. We implemented a greedy heuristic to find an *assembly* of the reads using both the edge weights and the extra information of the mapping locations of the other mates.

Merging the OEA and orphan contigs

Given the set of OEA contigs and the orphan contigs, we would like to find the maximum number of orphan contigs that can be merged with OEA contigs. We do not allow an orphan contig to merge with OEA contigs of both strands (say oea_+ and oea_-) if the score of the prefix/suffix alignment between the two ends of the orphan contig and oea_+ and oea_- is less than a user-defined threshold. We mathematically model this problem as a maximum-weight bipartite matching problem and give an exact solution based on the Hungarian method.

Let $Orp_{co} = \{or_1, or_2, \cdots, or_k\}$ be a set of orphan contigs and $OEA_{co} = \{oea_1, oea_2, \cdots, oea_v\}$ be a set of OEA contigs where oea_i is a pair of two OEA contigs from the local assembly of the OEA cluster with id i. [i.e., $oea_i = (oea_{i_F}, oea_{i_R})$]. We would like to assign each element of Orp_{co} (e.g or_i) to an element in OEA_{co} (e.g. oea_j) such that the total score of alignment of suffix of or_i with oea_{j_F} and the score of alignment of prefix of or_i with oea_{j_R} is maximized.

We reduce this problem to the maximum-weight matching problem in the bipartite graph $G(U, V, E)$ where G is defined as follows:

- $\forall or_i \in Orp_{co} : \exists u_i \in U$
- $\forall oea_j \in OEA_{co} : \exists v_j \in V$
- The weight of edge (u_i, v_j) is a function of the overlap between the the first $\Delta_{\max}$ base-pair of or_i and oea_{j_F} and the overlap between the last $\Delta_{\max}$ base-pair of or_i with oea_{j_R}.

Figure 2 shows how we reduce the merging problem into a bipartite matching problem.

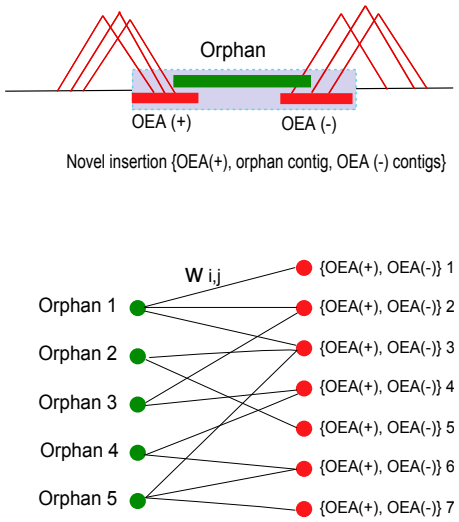


Figure 3.2: This figure illustrates how to reduce the problem of merging the orphan contigs with OEA contigs (note that each OEA cluster is in fact two contigs with different orientations, which together represent an insertion) into a maximum weighted matching problem in bipartite graphs. Each orphan contig is represented as a green node and each pair of OEA contigs (the OEA contigs of end-reads with $'+'$ and $'-'$ orientation mapping in the same OEA cluster) are represented as red nodes. The edge weights are the total alignment suffix/prefix score between the two OEA contigs (different orientations) and the orphan contigs.

3.1.3 Experimental results

We tested our framework using the whole genome shotgun (WGS) sequence library generated from the genome of an anonymous Yoruba African donor (NA18507) generated with the Illumina Genome Analyzer platform [21]. The genome of NA18507 has been previously studied by many groups including us [55, 7] to discover structural variation and copy number polymorphism. This dataset contains approximately 3.5 billion sequence reads ($\sim$ 1.7 billion pairs) of length $36 - 41$bp with an $InsSize$ of $\sim$ 209bp [21, 55]. $InsSize$ distribution of this dataset is shown in [55].

Preprocessing. Similar to the pre-screeing methodology used in [55], we removed any paired-end reads from consideration if either (or both) end sequence has an average *phred* [38] quality value less than 20, or if either (or both) sequence contain more than 2 unknown (i.e. N) characters.

Mapping onto the reference genome. After the preprocessing step, we mapped all the remaining $\sim$ 2.2 billion end sequences to the human genome reference assembly (UCSC build 36) using

mrFAST [7], allowing for edit distance ≤ 2. Note that mrFAST returns *all* possible map locations of read sequences, thus an OEA read can be aligned to multiple locations in the reference genome. In total, $15,173,562$ pairs of reads ($30,347,124$ end-sequences) were identified as *orphans*, while $83,662,790$ reads were identified as *OEAs*.

Orphan assembly. Using ABySS [134], we assembled the orphan paired-end reads into $4,154$ contigs of size $\geq$200bp. (N50 = 995). In the rest of this chapter we call these contigs as *ABySS contigs*. As an independent assessment, we also generated the sequence assembly of the orphans using the EULER [28] algorithm, which we call *EULER contigs*. EULER returned $4,564$ contigs of size $\geq$200bp. (N50 = 730).

Contamination removal. Next, we screened the orphan contigs to test for contamination. Using BLAST [12], we compared the orphan contigs with the `nt` database [4], and removed the contigs that align to consensus sequences of known contaminants (Escherichia coli, bacteriophage, herpesvirus, plasmid, Epstein-Barr, bacterium, bacteria) from further consideration. In total, 39 contigs were removed from the ABySS contig set due to contamination, where the majority were due to Epstein-Barr, a virus commonly used for cell immortalization. Figure 3.3 shows the length distribution of the ABySS contigs of length $\geq$100bp after the contamination removal. Note that out of 4,115 ABySS contaminant-free contigs ($\geq$200bp), 1,984 are $\geq$500bp and 778 are $\geq$1Kbp long. Among the EULER contaminant-free contigs, $1,690$ are $\geq$500bp and 582 are $\geq$1 Kbp.

We then mapped the orphan contigs to the human genome reference assembly (both build35 and build36) using BLAST in order to remove the orphan contigs with high sequence identity with the reference genome. 493 of ABySS contigs of length $\geq$200bp could be mapped onto either build35 or build36 with more than 99% identity (548 of EULER contigs). We removed such contigs from consideration in the remainder of the NovelSeq pipeline. The observation that 493 of the ABySS contigs could be aligned to the reference genome (build35 or build36) can be explained as follows: The orphan reads used in creating the ABySS contigs might be from regions of the genome with a high abundant of SNPs or short indels and could not be mapped onto the reference genome with a high similarity (edit distance ≤ 2). However, when using BLAST, the assmebly of those orphan reads could be mapped onto the reference genome. Another reason could be that such contigs were generated from the reads classified as orphans due to a low-quality sequence at the tails of the reads. They can still be assembled into reliable contigs since both AbySS and EULER built de Bruijn

[4] `http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastdb.html`

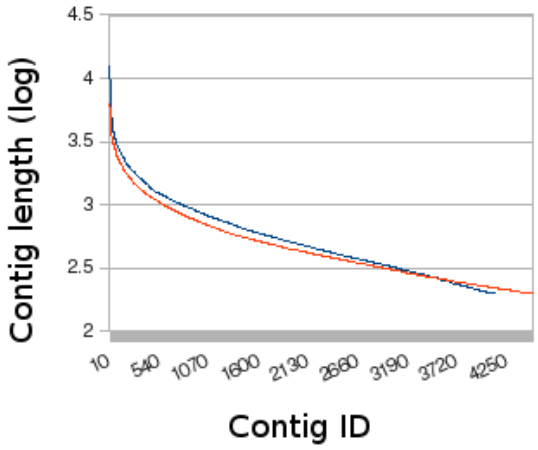


Figure 3.3: Histogram of the length distribution (log scale) of all ABySS (blue) and EULER (red) contigs of at least $200bp$ long.

graphs from 25bp subsequences of the reads, effectively discarding the sequence tails causing the mapping artefacts.

We used our clustering algorithm followed by the set cover approach to cluster the OEA reads, and obtained $10,560$ sets of OEA clusters with a *high support* [5] on each side (i.e. both + and - strands). Each side (or strand) of the detected OEA clusters were independently assembled with our local assembly routine, mrSAAB. Resulting OEA contigs were then processed together with the orphan contigs in the last phase of the NovelSeq pipeline, mrBIG. In summary, we anchored 130 EULER contigs and 113 ABySS contigs independently to the reference genome using the NovelSeq pipeline. In the merging phase of the orphan and OEA contigs (mrBIG), NovelSeq requires the alignment score between the orphan contig and the OEA contig to be $\geq$50. The alignment score is calculated as the score of the local alignment under affine gap model, where the *match* score is $+1$, *mismatch* penalty is -1, and *gap* penalties are -16 and -4 for gap opening and extension, respectively. The minimum requirement for the alignment score is a user defined parameter in the NovelSeq pipeline. Clearly, the lower alignment score one chooses at the merging phase, the more

[5] We considered the OEA clusters supported by $\geq$10 OEA reads in both strands, where $\geq$20 OEA reads were required to support the cluster in at least one strand.

orphan contigs can be anchored to the reference assembly.

Recently, Kidd et al. sequenced all fosmid clones (∼40Kbp each) generated from the genome of the same individual (NA18507) using the traditional Sanger method to build a map of novel insertions with high quality sequence information [75]. We used this dataset as the gold standard to test the accuracy of the NovelSeq pipeline. As shown in Table 3.1, we anchored >70% of the orphan contigs (with high sequence identity to a novel insertion sequence detected by fosmids) to locations in concordance with the fosmid-based predictions. Our concordance rate increases to 78% for ABySS contigs of length ≥500bp. Note that some of the fosmid sequences were not anchored to the human genome reference assembly, thus we were not able to test the accuracy of the loci we predicted for the contigs that are highly identical to such fosmid sequences.

NA18507	# merged		same locus		different locus	
minimum length	$500bp$	$200bp$	$500bp$	$200bp$	$500bp$	$200bp$
ABySS	78	113	37	50	10	21
EULER	85	130	35	51	14	23

Table 3.1: This table shows two different result sets depending on the minimum length of the orphan contigs considered for the merging phase. For both AbySS and EULER contigs, we show the number of orphan contigs that are merged with an OEA contig (and hence anchored) with an alignment score ≥50. *Same locus* (table header) indicates the number of orphan contigs with high sequence identity to a novel insertion sequence detected by fosmids and loci in concordance with the fosmid-based predictions. *Different locus* (table header) indicates the number of orphan contigs with high sequence identity to a novel insertion sequence detected by fosmids but with loci not in concordance with the fosmid-based predictions.

We need to re-emphasize that anchoring a novel insertion is not an easy task if there are repeat sequences (that also are not represented in the reference genome) at the flanks of the inserted sequence. Note that the dataset used here is generated by the Illumina platform and the insert size is very small (209bp). Any anchoring strategy that utilizes the OEA concept would fail to do so in such cases, since the OEA read pair will be too short to span over the flanking repeat if the repeat length is larger than the insert size (for example an *Alu* element is typically 300bp). For a more reliable OEA/orphan anchoring step, longer insert sizes are required.

Comparison of the orphan contigs with the NA18507 fosmid shotgun sequence library

We compare the sequence content of both ABySS and EULER contigs with a set of $2,509$ sequence contigs assembled from one end anchored fosmid end sequences as previously described by Kidd et

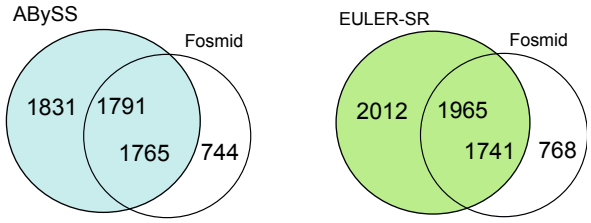
al. [75]. This fosmid resource was end-sequenced using capillary technology, and in the remainder of this chapter, we denote the sequence assembly generated from this dataset as *fosmid contigs*. Using cross_match [42] with default parameters, we observed that $1,789$ ($\sim$71%) fosmid contigs overlap with the ABySS contigs, and $1,754$ ($\sim$70%) fosmid contigs overlap with the EULER contigs. Figure 3.4(a) shows the comparison between ABySS and EULER contigs against the fosmid contigs. Next, we compared both ABySS and EULER orphan contigs with the novel sequences found by a recent study by Li et al. [92] (n=3,630; Figure 3.4(b)) based on whole-genome *de novo* sequence assembly using SOAPdenovo [92]. The reader can easily verify that *de novo* sequence assembly using the entire next-generation shotgun sequence read library requires extensive computational resources that are not needed by our method. Finally, Figure 3.4(c) depicts the comparison between ABySS and EULER contigs and the SOAPdenovo [92] and fosmid contigs.

Comparison with WGS libraries and the Venter genome

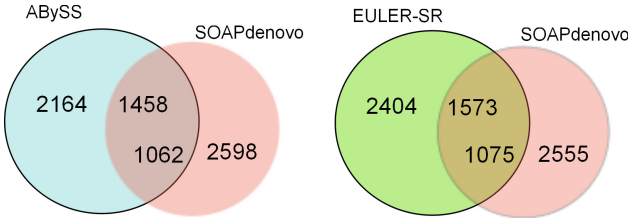
Finally, we used BLAST to compare the contaminant-free orphan contigs generated by ABySS (n=4,115) and EULER (n=4,525) with the WGS library generated from the genome of the same individual (NA18507) using Sanger sequencing, WGS library generated from the genome of Craig Venter [89], as well as the sequence assembly of the Venter genome (HuRef [89]. In Table 3 we also provide comparisons against human genome reference assembly (both build35 and build36). We required 99% and 95% sequence identity to call a hit in the database search. In addition, we provide the comparison statistics separated by the minimum contig length (i.e. $\geq$200bp and $\geq$500bp). We observe that the novel sequences detected in NA18507 genome are also found in the Venter genome, suggesting that these sequences correspond to rare deletions in the reference genome assembly.

3.2 Transposon insertion discovery

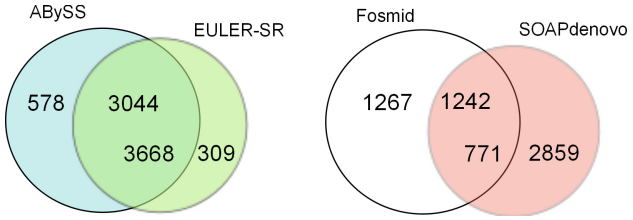
Transposons are repetitive elements in the genome that occupy a large fraction of the human genome, including *Alu*, L1, and SVA elements [107]. Most of these elements are fixed in the human lineage; however, around 0.05% of the transposons are still active, and the copy number and loci of these active transposons vary in the genomes of different individuals. Many studies have demonstrated that transpoable elements contribute to genome evolution and human genetic diversity. An interesting case was shown by Bekpen *el al.* [20]: insertions of an *Alu* element were posited to cause pseudogenization of the IRGM gene at the split of New World and Old World monkey lineages



(a) The comparison of ABySS and fosmid contigs (left), and the comparison of EULER and fosmid contigs [75] (right)



(b) The comparison of ABySS and SOAPdenovo contigs [92] (left), and the comparison of EULER and SOAPdenovo contigs (right)



(c) The comparison of ABySS and EULER contigs (left), and the comparison of SOAPdenovo [92] and fosmid contigs [75] (right)

Figure 3.4: Venn diagrams depicting pairwise comparisons of novel sequence assemblies generated by ABySS, EULER, SOAPdenovo [92], and fosmid end-sequences using *phrap*. Note that we provide two numbers at the intersections, corresponding to the numbers of contigs in each set that are almost identical to the contigs in the reciprocal set.

NA18507		$\geq 200bp$		$\geq 500bp$	
	database	95%	99%	95%	99%
ABySS	build35	616	481	236	174
ABySS	build36	611	475	222	159
ABySS	NA18507 (fosmid end-seq.)	2305	1944	1253	1076
ABySS	Venter WGS	3028	2938	1811	1798
ABySS	HuRef	3815	3763	1512	1488
EULER	build35	670	530	123	100
EULER	build36	660	522	114	92
EULER	NA18507 (fosmid end-seq.)	2530	2169	1055	933
EULER	Venter WGS	4193	4131	1542	1536
EULER	HuRef	3272	3127	1329	1309

Table 3.2: The comparison of NA18507 orphan contigs with the WGS libraries and the Venter genome. For different cases, the number of orphan contigs with a high similarity to each library is given.

35-40 million years ago (mya) by disrupting the open reading frame (ORF). A second transposon integration (ERV9) restored the ORF $\sim$24 mya in the common ancestor of apes and humans, demonstrating the first report on a "resurrected" gene. The human IRGM gene plays an important role in the immune system [20] and is associated with Crohn's disease [98]. Transpositions are associated with the expansion of interspersed segmental duplications [14] and can promote both the creation of segmental duplication in human genomes [157] and the alteration of gene transcription by gene-trapping and exonization.

The discovery of the *Alu* elements more than 30 years ago [130, 59] as 300 basepairs (bp) interspersed repeat sequences commonly found within the introns of genes [34] prompted an active area of research to address the role of transposable elements in genome evolution and human disease [17]. More than one million *Alu* retrotransposons comprise over 10% of the human genome sequence [4, 17, 61]. They are partitioned into numerous subfamilies, which have been active at different time points during primate evolution [117, 93]. Currently, 30 distinct categories of *Alu* subfamilies are recognized [107] with AluYa5 and AluYb8 being most active in the human lineage [26]. *Alu* retrotranspositions have numerous consequences leading to insertional mutations, gene conversion, recombination, alterations in gene expression, pseudogenization, structural variation and formation of segmental duplications [17, 14, 69, 157].

Traditional methods to detect *Alu* insertion polymorphisms involve polymerase chain reaction (PCR) where putative polymorphic loci are genotyped one by one [15, 128, 32]. Recently, PCR-based capture and high-throughput sequencing methods have been applied to quickly screen thousands of transposition events [37, 155]. Although promising, these methods also require the design

of appropriate PCR primers and are susceptible to cloning failures. Other methods to detect retrotransposons include paired-end and full fosmid sequencing [75, 19, 76], transposon insertion profiling by microarray [60], and restriction enzyme profiling followed by Sanger and Roche 454 sequencing [65]. Whole-genome shotgun sequencing (WGS) of different individuals [89, 21, 152, 154, 101] provides a resource to discover *Alu* element insertions at a much higher scale and throughput. However, such findings are limited by the read length of the sequencing platform [157], and few studies have attempted to systematically discover these events at the individual genome level.

In what follows, we describe a computational method to discover transposon insertions in genomes sequenced by paired-end next-generation sequencing (NGS) platforms. We demonstrate the sensitivity and specificity of our algorithm by simulation, proving its detection power. We also apply this algorithm to construct *Alu* retrotransposition maps from the genomes of eight human individuals sequenced with the Illumina platform. In addition, we also analyze one Yoruban trio from Ibadan, Nigeria, and describe the properties of parent-to-child *Alu* transmission.

3.2.1 The formulation of transposition events

In the first chapter, we discussed the general framework for detecting structural variation events using paired-end reads, based on aligning the paired-end reads to the reference genome and observing the discordancies. For a full case study of deletion, small insertion and inversion events, we refer the reader to [55]. In the previous section, we demonstrated framework to identify novel sequence insertions, using paired-end sequencing. Here, we focus on identifying transposons in sequenced genomes.

As usual, we denote a read pair as pe_i and the distance between the end coordinate of the proximal read and the start coordinate of the distal read as the *GapSize* (i.e. the *insert-size* minus the total length of the reads). An alignment of a read pair to the reference genome is denoted as *concordant* [147] if the distance between the aligned end-reads is in the range $[\Delta_{\min}, \Delta_{\max}]$ and the alignment orientation is $+-$. The range $[\Delta_{\min}, \Delta_{\max}]$ is empirically calculated by analyzing the mapping span size distribution of the read pairs.

The set of discordant paired-end reads is represented as $R = \{pe_1, pe_2, \cdots, pe_n\}$. Each discordant paired-end read pe_i may be mapped to multiple locations in the reference genome. The set of all alignments of pe_i is then defined as $Align(pe_i) = \{a_1 pe_i, a_2 pe_i, \cdots, a_j pe_i\}$.

Note that each alignment of pe_i to the reference genome ($a_j pe_i$) is a 5-tuple that represents the

map locations of the end-reads and their alignment orientation. More formally,

$$a_j pe_i = ((L_\ell(pe_i), L_r(pe_i)), (R_\ell(pe_i), R_r(pe_i)), or(pe_i))$$

where the pair $(L_\ell(pe_i), L_r(pe_i))$ represents the map location (i.e. both start and end loci) of the left end-read of pe_i, $(R_\ell(pe_i), R_r(pe_i))$ is the mapping location of the right end-read of pe_i, and $or(pe_i)$ represents the map orientation of both ends. Note that $or(pe_i) \in \{+-, ++, --, -+\}$.

In what follows, we study two classes of transposition events and present the set of conditions based on the map locations and orientations of the paired-end alignments that imply a transposition event within each of these classes. First, we consider those transposition events in which the transposed segment is in direct orientation, and present the set of conditions for all of the four different cases of this class (denoted as Class I). We denote this type of transposition event by $SV_{Copy}(Loc_L, Loc_R, Loc_{Br_L}, Loc_{Br_R})$. This indicates a region being copied is a segment inside loci $[Loc_L, Loc_R]$ (i.e. a substring of region $[Loc_L, Loc_R]$), and it is pasted (copied) to a locus between Loc_{Br_L} and Loc_{Br_R}. We also study the cases for Class II, where the transposon is copied in inverted orientation, and we denote the event as $SV_{\overline{Copy}}(Loc_L, Loc_R, Loc_{Br_L}, Loc_{Br_R})$.

A transposition SV of Class I is defined as $SV_{Copy}(Pos_L, Pos_R, Pos_{Br})$, where the genomic segment from positions Pos_L to Pos_R is copied into location Pos_{Br}. Similarly, a copy event $SV_{\overline{Copy}}$ denotes a transposition event in inverted orientation.

One of the following four cases should hold for a paired-end read alignment ape that supports a transposition event (Class I):

Case 1 ($Pos_{Br} < Pos_L$ and $or(ape) = +-$) : $\Delta_{\min} < Pos_{Br} - L_r(ape) + R_\ell(ape) - Pos_L < \Delta_{\max}$
(Figure 3.5(a))

Case 2 ($Pos_{Br} < Pos_L$ and $or(ape) = -+$) : $\Delta_{\min} < L_\ell(ape) - Pos_{Br} - R_r(ape) + Pos_R < \Delta_{\max}$
(Figure 3.5(b))

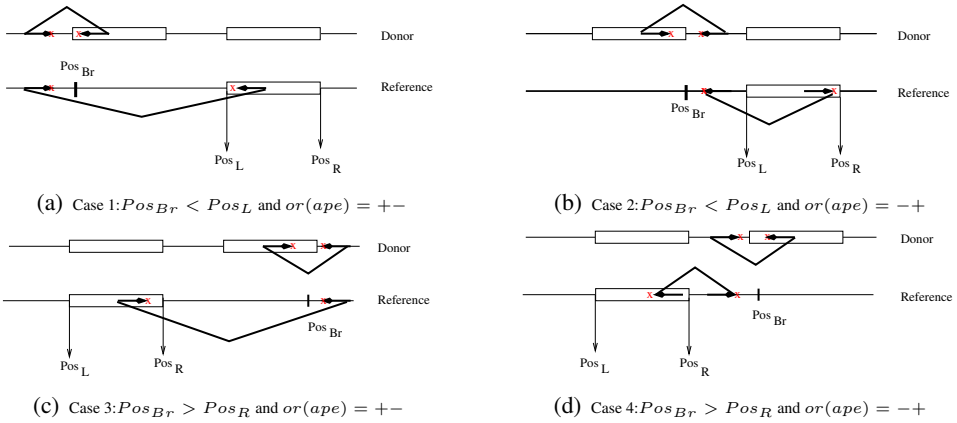
Case 3 ($Pos_{Br} > Pos_R$ and $or(ape) = +-$) : $\Delta_{\min} < R_\ell(ape) - Pos_{Br} + Pos_R - L_\ell(ape) < \Delta_{\max}$
(Figure 3.5(c))

Case 4 ($Pos_{Br} > Pos_R$ and $or(ape) = -+$) : $\Delta_{\min} < Pos_{Br} - R_r(ape) + L_\ell(ape) - Pos_L < \Delta_{\max}$
(Figure 3.5(d))

Similarly, one of the following cases should hold for a transposition event of Class II:

Case 1 ($Pos_{Br} < Pos_R$ and $or(ape) = ++$) : $\Delta_{\min} < Pos_{Br} - L_r(ape) + Pos_R - R_r(ape) < \Delta_{\max}$
(Figure 3.6(a))

Case 2 ($Pos_{Br} < Pos_R$ and $or(ape) = --$) : $\Delta_{\min} < L_\ell(ape) - Pos_{Br} + R_\ell(ape) - Pos_L < \Delta_{\max}$
(Figure 3.6(b))



(a) Case 1:$Pos_{Br} < Pos_L$ and $or(ape) = +-$

(b) Case 2:$Pos_{Br} < Pos_L$ and $or(ape) = -+$

(c) Case 3:$Pos_{Br} > Pos_R$ and $or(ape) = +-$

(d) Case 4:$Pos_{Br} > Pos_R$ and $or(ape) = -+$

Figure 3.5: The set of conditions for each case that suggests a transposition event in which the transposed segment is copied in direct orientation (Class I).

Case 3 ($Pos_{Br} > Pos_R$ and $or(ape) = ++$) : $\Delta_{\min} < Pos_{Br} - R_r(ape) + Pos_R - L_r(ape) < \Delta_{\max}$
(Figure 3.6(c))

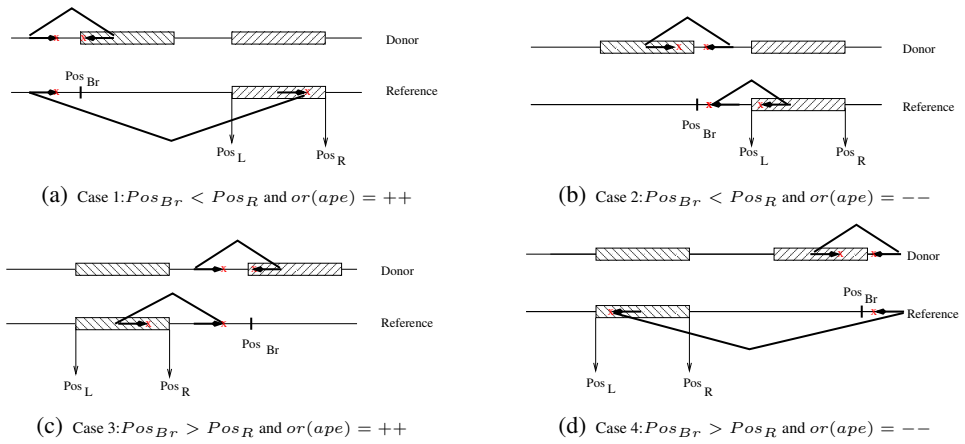
Case 4 ($Pos_{Br} > Pos_R$ and $or(ape) = --$) : $\Delta_{\min} < R_\ell(ape) - Pos_{Br} + L_\ell(ape) - Pos_L < \Delta_{\max}$
(Figure 3.6(d))

Maximal valid clusters and transposition event detection

A set of discordant paired-end read alignments that support the same potential transposition event is called a "valid cluster" and denoted by $VClu_i = \{a_{i'_1}pe_{i_1}, a_{i'_2}pe_{i_2}, \cdots, a_{i'_\ell}pe_{i_\ell}\}$.

As per [55], a "maximal valid cluster" is defined as a valid cluster where no additional paired-end read alignments can be added such that the cluster remains valid. Note that all paired-end read alignments in maximal valid clusters suggest the same potential structural variation. It was shown that it is sufficient to calculate all maximal valid clusters to solve the maximum parsimony structural variation (MPSV) problem. This can be done in polynomial time (with respect to the number of paired-end alignments), where the computation of *all* valid clusters is unnecessary, and exponential in run time. In [55, 136], efficient algorithms to find the maximal valid clusters are presented to predict insertion, deletion, and inversion events.

To find all maximal valid clusters for transposition events, a naïve method would investigate all $O(n^3)$ possibilities of potential transposition events, for each of the locations Pos_{Br}, Pos_L, and Pos_R between 1 an n, where n is the genome length. This can be done by first creating a cluster



(a) Case 1:$Pos_{Br} < Pos_R$ and $or(ape) = ++$

(b) Case 2:$Pos_{Br} < Pos_R$ and $or(ape) = --$

(c) Case 3:$Pos_{Br} > Pos_R$ and $or(ape) = ++$

(d) Case 4:$Pos_{Br} > Pos_R$ and $or(ape) = --$

Figure 3.6: The set of conditions for each case that suggests a transposition event in which the transposed segment is copied in inverted orientation (Class II).

for each possible values of Pos_{Br}, Pos_L, Pos_R, and then adding those paired-end reads that satisfy the conditions given above to the appropriate cluster. Finally, a set of maximal clusters would be selected. The above method guarantees to find all the maximal valid clusters but it would be time consuming in practice. In what follows, we will present a more efficient method to find all the maximal valid clusters provided that the potential positions of copied segments or copied sequences are known.

We define $\Phi = \{(\phi_{1_\ell}, \phi_{1_r}), (\phi_{2_\ell}, \phi_{2_r}), \cdots, (\phi_{t_\ell}, \phi_{t_r})\}$ as the set of (non-overlapping) segments that can be copied to other locations (Φ can represent the annotated transposons in the reference genome assembly). Note that $\forall\ i \leq t$, ϕ_{i_ℓ} is the start location of the i-th segment and ϕ_{i_r} is the end location. The coordinates for the intervals are referred to as "end-points" in the rest of this section for simplicity.

For each paired-end read mapping ape with exactly one end-read mapped to a transposon (e.g. $\phi_i = (\phi_{i_\ell}, \phi_{i_r})$), there exists a range of locations, or "breakpoint intervals" $Br^i(ape) = [Br^i_L(ape), Br^i_R(ape)]$, where ape supports a copy of subsequence $\phi_i = (\phi_{i_\ell}, \phi_{i_r})$ to any location within $Br^i(ape)$ in the reference genome. Note that for a given ape and a segment $\phi_i = (\phi_{i_\ell}, \phi_{i_r})$, the breakpoint interval $Br^i(ape)$ can easily be computed using the set of conditions given earlier in this section.

Now we present an efficient algorithm to find the maximal valid clusters supporting copy of segment $\phi_i = (\phi_{i_\ell}, \phi_{i_r})$ to any location in genome (the algorithm can trivially be extended for other

segments in Φ).

Without loss of generality we assume that there are total of m_i discordant mappings with exactly one end mapped to ϕ_i (Please note that $m = \sum_{i=1}^{t} m_i$, where total number of segments is t). Lets assume WLOG the set of such discordant paired-end read mappings is $Ape^i = \{ape_1, ape_2, \cdots, ape_{m_i}\}$. In addition the set of breakpoint intervals for the paired-end read alignments in set Ape^i is denoted as $Br^i = \{[Br^i_L(ape_1), Br^i_R(ape_1)], [Br^i_L(ape_2), Br^i_R(ape_2)], \cdots, [Br^i_L(ape_{m_i}), Br^i_R(ape_{m_i})]\}$.

It is trivial to see that finding maximal valid clusters for all copies of segment $\phi_i = (\phi_{i_\ell}, \phi_{i_r})$ into any position in the genome is equivalent to finding all maximal intersecting breakpoint intervals for all discordant paired-end read mappings ape^i, which have one end in the segment ϕ_i. Thus, we are interested in finding *all maximal intersections* of the breakpoint intervals Br^i. We present an $O(m \log m) + O(\text{size of output})$ algorithm which outputs all the maximal intersecting intervals. This algorithm is the optimal solution to this problem.

Algorithm for finding all the maximal intersecting intervals

Given a set of intervals (with cardinality of m_i), we want to find all the maximal intersecting intervals. We first sort all the end points of the m intervals ($2m$ endpoints) in ascending order based on their values. We call this sorted list of intervals L where each interval appears twice in the list L. We then scan the sorted list from left to right:

- If we observe a point which is at the left endpoint of an interval, we insert the interval to a minimum heap data structure, denoted as $heap1$. The priority value of $heap1$ is the right endpoint of the interval inserted to it. After each insertion of a new interval to $heap1$, we assign a value *true* to a flag denoted as $newIns$.

- If we observe a point which is at the right endpoint of an interval, we first check the flag $newIns$. If it isset to *true*, we output all the elements in $heap1$ as one maximal cluster and set $newIns$ to *false*. We then remove the interval from $heap1$ since it is guaranteed that the value of the right endpoint of the interval removed from the heap is the same as the right endpoint of the interval reached in scanned list L). We continue removing intervals from $heap1$ until the priority value of the head element of the heap remains unchanged.

The above algorithm outputs all the maximal intersecting intervals, which are indeed equivalent to the maximal valid clusters we were originally looking for.

Complexity : It can be shown that the running time of the above algorithm is $O(m_i \log m_i + s)$, where s is the size of output. The first step of the algorithm takes $O(m_i \log m_i)$, because of the sorting procedure. In the worst case, since each removal/insertion operation in the heap takes $O(\log m)$ time, the second step also takes $O(m \log m)$. It takes $O(s)$ to write the output and generate the maximal clusters. In addition, it is proven that even finding the maximum clique in an interval graph has a lower bound of $\Omega(m_i \log m_i)$ [45]. Thus, our algorithm gives the optimal solution for finding all maximal clusters.

Detection of transposon insertion events from maximal valid clusters

Each maximal valid cluster for different SV types (i.e. insertion, inversion, deletion or transposition) only suggests a *potential* structural variation event. The fact that each paired-end read can be mapped to multiple locations that are included in multiple maximal valid clusters proves that some of these implied potential variants are incorrect (i.e. some of the variations implied by maximal valid clusters are not real).

In [55], a combinatorial method was presented to select the minimum number of these clusters (which is equivalent to selecting the minimum number of structural variant events), such that each discordant paired-end read has a mapping in to at least one of the selected valid clusters. This problem was called Maximum Parsimony Structural Variation (MPSV), and an approximation algorithm was given. A similar algorithm to the one in [55] can be used to find transposon insertion events from the maximal valid clusters created by the algorithms presented in the previous section.

In the next chapter, we will introduce an extension to Maximum Parsimony Structural Variation problem, and provide an efcient method to solve it. We will show that our new method outperforms our previous algorithm in [55].

3.2.2 Experimental Results

Implementation

It is well-known that the number of transposons in the human genome is quite large (around 1 million for only *Alu* elements). Considering all known transpoable elements (segments in genome) as potential transposon sequences for our algorithm would be very time consuming and unnecessary. These transpoable elements can be clustered together as super-families; in fact, most of these families are well known and their consensus sequences are available. In order to make experiments more manageable and less time-consuming, we used the consensus sequence of these transposon families

(available from [137]) to represent each of the families. We create a new sequence, denoted as chrN, by appending a poly N sequence to each of the consensus sequences from [137] and connecting them to each other. In our experiments only *Alu* and SVA were considered.

For mapping purposes, we use mrsFAST [46], a cache-oblivious short read mapper which was recently developed as an extension of mrFAST [7]. mrsFAST maps the paired-end reads to *all* locations with Hamming distance less than a user-defined threshold ω. In this experiment, we set $\omega = 2$.

Before running our algorithm, we find the mappings needed for our transposon insertion discovery algorithm in an efficient manner. Given paired-end short reads $R = \{pe_1, pe_2, \cdots\}$, these are the steps needed to find the mappings:

- First, we map all the paired-end reads to chrN via mrsFast and we discard the paired-end reads if none of their end-reads map onto chrN. The remaining reads are the ones that have at least one end mapped to chrN.
- Second, after all the reads returned in the first step are mapped to the reference genome, we discard all the the paired end reads that have at least one *concordant* mapping.
- Third, we map the reads from step two as single-end reads to chrN and to the reference genome. We post-process these mappings to get all the paired mapping locations that have one end in chrN and the other end in the reference genome. The outcome of this step will be used as the input mapping for our transposon algorithm described earlier.

Transposon insertion discovery on J.Craig Venter genome

A list of transposon insertions into the J. Craig Venter genome (HuRef) [89] in comparison to Build 36 of the NCBI reference genome was published [157]. We used the available HuRef genome to produce short paired-end reads, very similar in theory to reads which are generated by the Illumina technology (assuming Venter genome was sequenced by Illumina platform), to see how many of the known/validated mobile insertions our algorithm is able to predict and test how many of the known mobile insertions it will not find.

In our experiment, we examined our transposon insertion discovery algorithm on 22 chromosomes of HuRef (from chr1 to chr22). We created paired-end reads from the HuRef genome with a read length of 36bp and coverage of 10x-fold. The fragment insert-sizes for paired-end reads were chosen randomly from a normal distribution very similar to the fragment size distribution of

Chromosome	Validated	Predicted	Found	Recall	Precision
Chromosome 1	41 *Alu* 4 NCAI	42 *Alu*	40	88%	95%
Chromosome 2	59 *Alu* 2 NCAI	57 *Alu*	56	91%	98%
Chromosome 3	42 *Alu* 1 NCAI 1 SVA	40 *Alu*	40	90%	100%
Chromosome 4	43 *Alu* 4 NCAI	41 *Alu* 1 NCAI	40 1	87%	97%
Chromosome 5	39 *Alu* 1 SVA	44 *Alu* 1 SVA	35 1	90%	80%
Chromosome 6	59 *Alu* 1 NCAI 1 SVA	55 *Alu* 1 SVA	53 1	88%	96%
Chromosome 7	24 *Alu*	22 *Alu*	20	83%	91%
Chromosome 8	34 *Alu* 2 NCAI	33 *Alu*	33	92%	100%
Chromosome 9	23 *Alu* 1 NCAI 1 SVA	23 *Alu* 1 NCAI	21 1	88%	92%
Chromosome 10	33 *Alu* 2 NCAI	32 *Alu* 1 NCAI	32 1	94%	100%
Chromosome 11	35 *Alu* 3 NCAI 2 SVA	32 *Alu* 1 NCAI 1 SVA	32 1 1	85%	100%
Chromosome 12	33 *Alu* 4 NCAI	34 *Alu*	31	84%	91%
Chromosome 13	34 *Alu* 3 NCAI 2 SVA	34 *Alu* 1 SVA	34 1	90%	100%
Chromosome 14	19 *Alu* 1 NCAI 2 SVA	20 *Alu* 2 SVA	19 2	95%	95%
Chromosome 15	24 *Alu*	21 *Alu*	20	83%	95%
Chromosome 16	12 *Alu* 3 NCAI	12 *Alu* 1 NCAI	12 1	80%	100%
Chromosome 17	9 *Alu* 2 NCAI 2 SVA	9 *Alu* 1 SVA	9 1	77%	100%
Chromosome 18	22 *Alu*	21 *Alu*	20	91%	95%
Chromosome 19	11 *Alu* 3 NCAI 1 SVA	11 *Alu* 1 NCAI	11 1	80%	100%
Chromosome 20	11 *Alu* 2 NCAI	13 *Alu* 1 NCAI	9 1	77%	71%
Chromosome 21	7 *Alu*	7 *Alu*	7	100%	100%
Chromosome 22	7 *Alu*	5 *Alu*	5	71%	100%

Table 3.3: In this table we show the precision and recall of our transposon (*Alu*, NCAI, and SVA) insertion discovery algorithm on Venter Genome. Our algorithm is run on simulated short paired-end reads created from the assembled genome [89]. The comparison is done against the *Alu* insertions found and validated by [157] using the full assembled genome. As it can be seen in all the chromosomes that we experimented, impressively our algorithm has a very high recall and precision.

NA18507 sequenced genome published by the Illumina technology [21]). We used simulated short paired-end reads form HuRef genome, instead of using the real reads from NA18507 individual, because the list of validated mobile insertion elements for this individual is not yet known. See Table 3.2.2 for the results.

In our experiments we tried to recover *Alu*, NCAI (Non-classical *Alu* Insertion), and SVA insertions. In Table 3.2.2, our results are compared with known transposon insertions on the HuRef genome published in [157]. As you can see, our method is able to find most of the known/validated transposon insertions (high recall rate - more than 85%) while the number of the predicted calls that do not match the list of validated calls published in [157], is very low (high precision rate around 90%).

We also have made an interesting observation about the *Alu* insertions (*Alu* and NCAI) which were not found by our algorithm. Most of the *Alu* insertions which were not found by our algorithm, were much shorter than the set of *Alu* consensus sequences we used (the set of of *Alu* consensus sequences is provided in [137]) in creating chrN for our mappings. We show the distribution of length of *Alu* insertions which we were not able to find is very different from the length of the *Alu* consensus sequences provided in [137]. This suggests that majority of the *Alu* insertions which were not found by our algorithm is because of the lacking of their consensus sequence in chrN, not the shortcoming of our discovery algorithm.

Figure 3.7 shows the rate of false and true discovery of our algorithm for differnet length ranges of the *Alu* sequences. As it can be seen in the figure, the rate of true discovery of our alagorithm for full-length *Alu* elements is near perfect.

Alu insertion discovery on 8 human whole genomes

We downloaded WGS data (http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi) from the genomes of eight human individuals generated using the Illumina paired-end sequencing technology (Table 3.4). We considered individuals from different populations, including three Yoruban individuals from Ibadan, Nigeria (YRI: NA18506, NA18507, and NA18508 [21]), one Centre d'Etude du Polymorphisme Humain (CEPH) individual of European origin (Utah resident with ancestry from northwestern Europe, CEU: NA10851 [111]), two Khoisan individuals from southern Africa (KB1 [131] and HGDP01029 [43]), one Han Chinese (YH [152]), and one Altaic Korean (AK1 [78]). The three Yoruban genomes constitute a parent-child trio providing us the opportunity to study transmission of *Alu* insertions (Table 3.4). We computationally predicted novel *Alu* insertion loci using the method we presented earlier in this section [57]. Briefly, we mapped the WGS data using mrFAST [7] to the

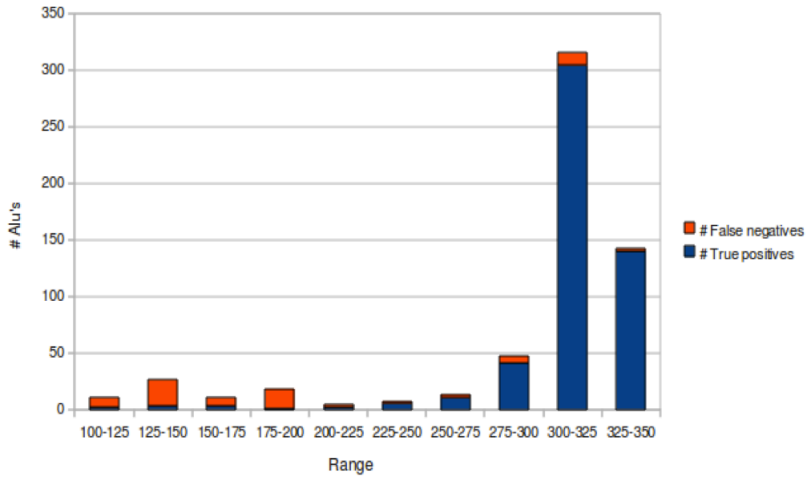


Figure 3.7: The length distribution of true positive and false negative *Alu* insertion predictions. Note that all of the *Alu* consensus sequences used in creating chrN were longer than 250bp.

reference genome (National Center for Biotechnology Information (NCBI) build 36) and identified all discordant read pairs. We then realigned such reads to both the reference genome and a database of *Alu* consensus sequences using a modified version of mrsFAST [46]. We applied VariationHunter to predict *Alu* insertions in the sequenced samples dynamically adjusting the minimum read support as a function of sequence and physical coverage of each analyzed genome (Table 3.4).

In total, we predicted 2,913 novel *Alu* insertions not present in the human reference genome for the YRI trio sequence data (see Figure 3.8) and a total of 4,342 *Alu* insertions in the entire

Individual	Population	# reads (M)illion	Read Length (bp)	Insert size (bp)	Min Support	# *Alu*	dbRIP	dbRIP + Others
NA18506	YRI	3444M	35	222	6	1807	294	440
NA18507 [21]	YRI	2261M	36-41	208	6	1377	292	435
NA18508	YRI	3175M	35	203	6	1714	310	451
NA10851 [111]	CEU	1309M	36-101	132-384	5	1282	370	501
AK1 [78]	Korean	1430M	36-106	132-384	2	909	225	327
YH [152]	Han Chinese	979M	35	135-440	3	1160	307	462
KB1 [131]	Khoisan	842M	36-37	181	2	457	92	144
HGDP01029 [43]	Khoisan	161M	76	150-300	2	307	60	93

Table 3.4: Genome of eight donors studied for *Alu* insertions.

set (see Figure 3.9). We find that only 13.2% (571/4,342) of these loci have been previously reported in the database of retrotransposon insertion polymorphism (dbRIP) [67]. If we include two additional, published surveys [65, 155], we find that 79.0% (3,432/4,342) of our calls are novel (dbRIP+other column in table 3.4). Of the *Alu* integration sites, 33.1% (1,437/4,342) mapped within genes as opposed to the expected 37.3% of the genome based on the most current (RefSeq) gene definition (downloaded from University of California, Santa Cruz (UCSC) Genome Browser on May 20, 2010). This represents a significant ($p \leq 0.001$) depletion based on simulation confirming potential selection and preferential integration within gene-poor regions of the human genome (see Figure 3.10). We identified 31 *Alu* elements that retrotransposed within an exon, of which three are predicted to disrupt a coding sequence (please see Figure 3.11).

Our collaborators at the Genome Sciences department of the University of Washington, experimentally validated a set of *Alu* insertion predictions from seven of the eight genomes using PCR. The combined validation results suggest excellent sensitivity (63/64 = 98.4%) for our algorithm, but also suggest caution in interpreting the map location precision based strictly on in silico mapping. The detail of the PCR experiments are beyond the scope of this thesis and we refer the curious reader to the resource paper [56].

3.2.3 Familial Transmission

We focused on the parent-child trio (NA18506, NA18507 and NA18058) to assess the transmission characteristics of the novel *Alu* insertions. We treated each genome individually and then compared the genomes for unique and shared *Alu* retrotransposons. We found no significant difference between non-transmitted and transmitted *Alu* elements from either parent, although slightly more transmissions were predicted from the mother due to increased sequence coverage and X chromosome transmissions from the mother to the proband (NA18506) (please see Figure 3.12).

As it can be seen in the figure, the number of false-positive de novo events in the child is extremely high. We employed further heuristics on this data set, which led to the publication [56]. However, we do need further improved analysis to increase the sensitivity for transmitted *Alu* insertions. We will discuss new strategies for handling multiple genomes in the next chapter, in detail.

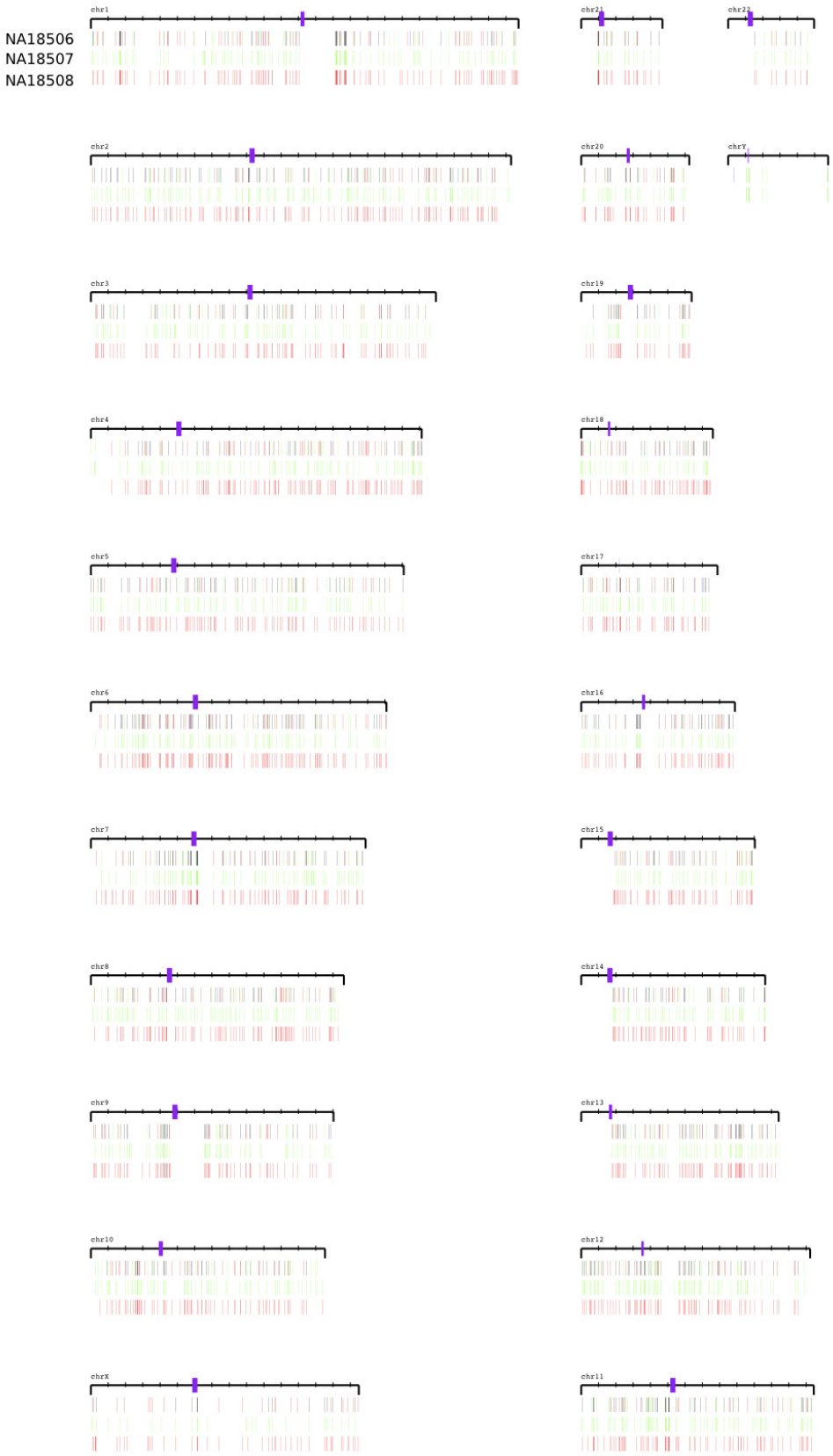


Figure 3.8: The *Alu* insertion loci are shown for the YRI trio in the order of NA18506, NA18507 and NA18508.

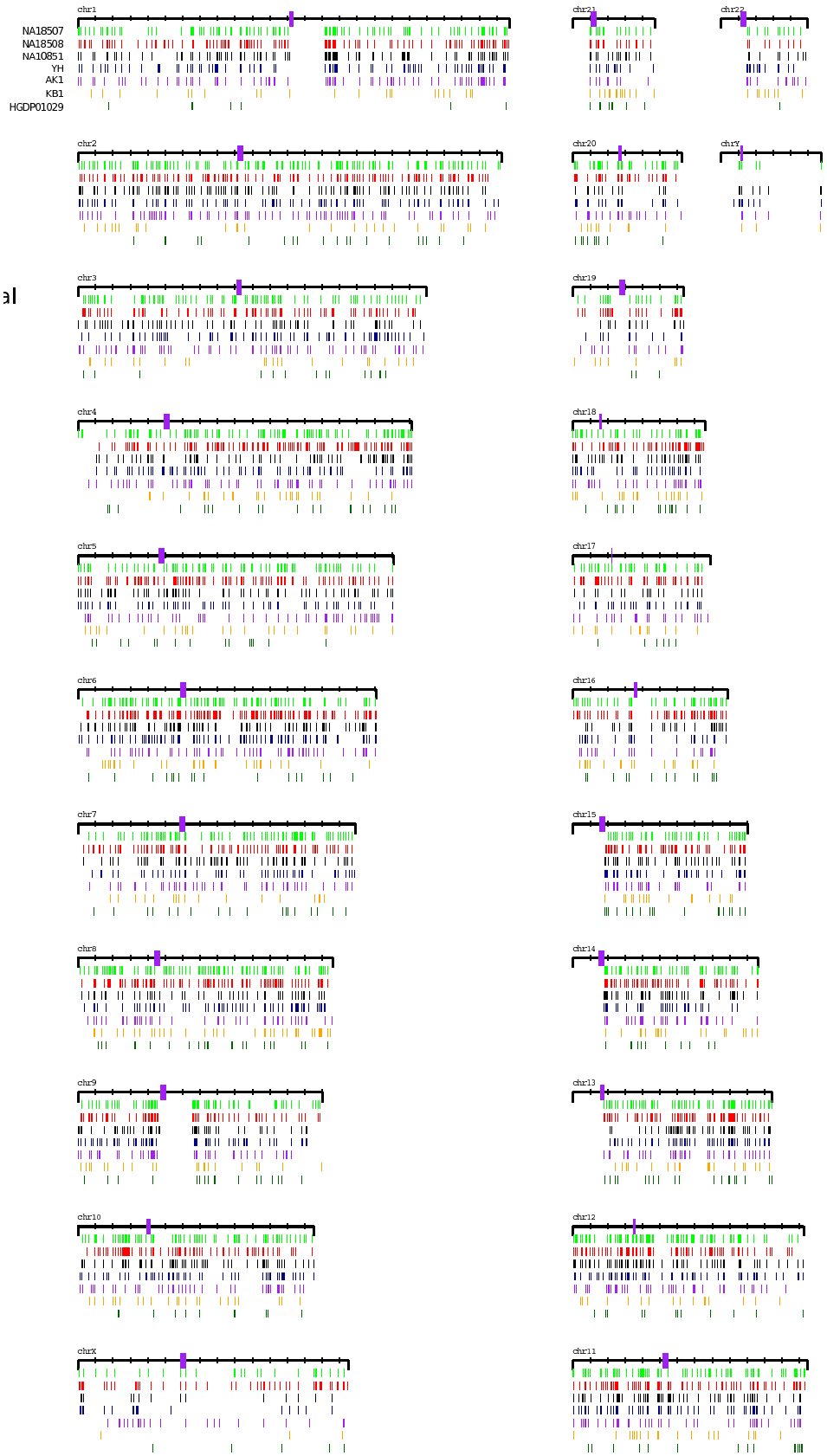


Figure 3.9: The *Alu* insertion loci are shown for 7 different individuals

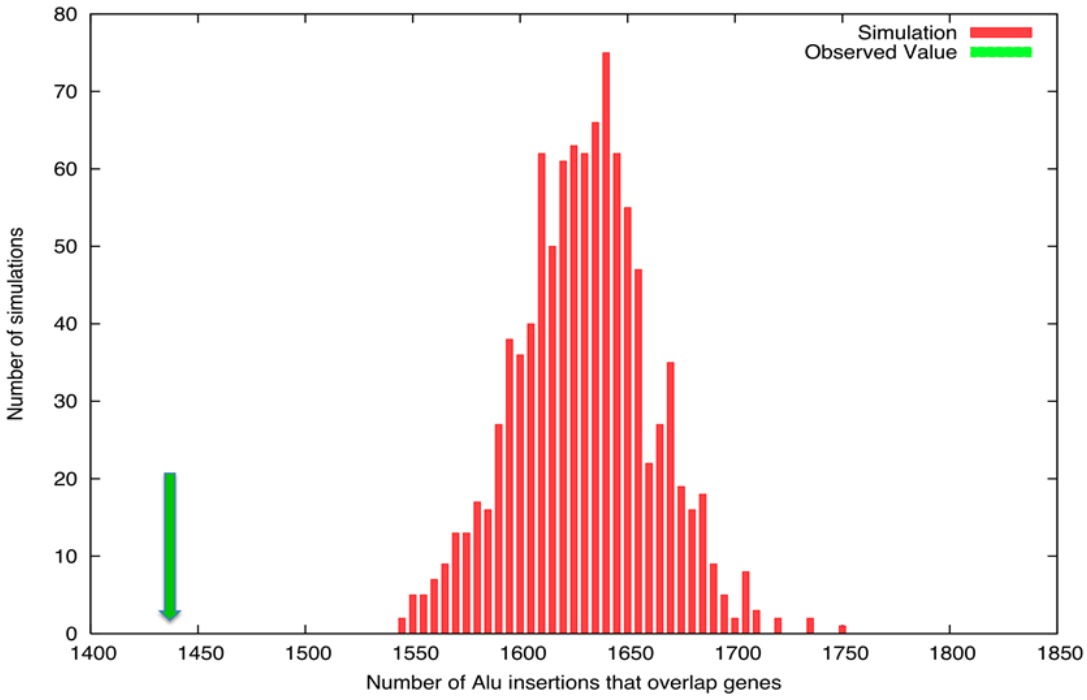


Figure 3.10: Gene overlap analysis. 1437/4342 (33.1%) of predicted *Alu* insertions map within a human gene as defined by RefSeq (green arrow). The histogram shows the expected distribution of gene overlap based on 1000 permutations.

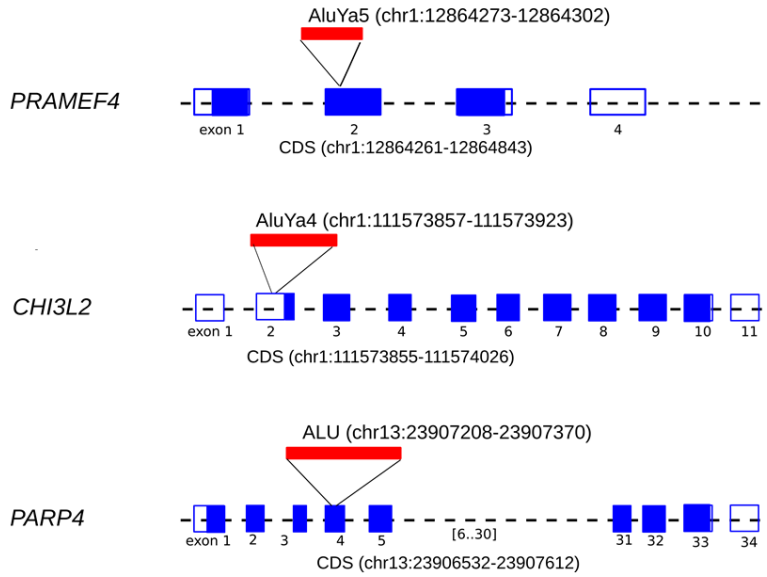


Figure 3.11: Gene disruptions. The locations of three novel insertions within the coding exons of PRAMEF4 (chr1:12,864,273-12,864,302), CHI3L2 (chr1:111,573,857-111,573,923), and PARP4 (chr13:23,907,208-23,807,370) are shown. Unfilled black rectangles represent the exons (and parts of exons) in the untranslated region (UTR), where filled rectangles show protein-coding exons. First and third figures are two predicted *Alu* insertions mapped within a coding region, while second figure is an insertion in UTR.

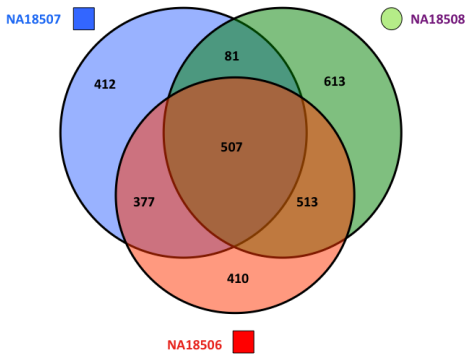


Figure 3.12: A three-way comparison of novel *Alu* insertion polymorphisms in the YRI trio: when they are predicted separately.

Chapter 4

Handling multiple genomes

With the increasing popularity of whole genome shotgun sequencing (WGSS) via high-throughput sequencing technologies, it is becoming highly desirable to perform comparative studies involving multiple individuals (from a specific population, race, or a group sharing a particular phenotype). The conventional approach for a comparative genome variation study involves two key steps: i) each paired-end high throughput sequenced genome is compared with a reference genome and its (structural) differences are identified, ii) the list of structural variants in each genome are compared against each other.

In order to identify genomic variations with much higher accuracy than what is currently possible, we propose to move from the current model of (1) detecting genomic variations in individual next generation sequenced (NGS) donor genomes independently, and (2) checking whether two or more donor genomes indeed agree or disagree on the variations - we will call this model, ”independent structural variation detection and merging” (ISV&M) framework. As an alternative, we introduce a new model in which genomic variation is detected among multiple genomes simultaneously.

Our new model can be likened to multiple sequence alignment methods which were introduced to overcome the limitations of pairwise sequence aligners - the primary source of sequence analysis in the early days of genomics. Pairwise sequence alignment methods implicitly aim to match identical regions among two input sequences which are not interrupted by mismatches or indels. They achieve this under the maximum parsimony principle that suggests to minimize the (probabilistically weighted) number of single nucleotide insertions, deletions and mismatches in an alignment. Unfortunately the most likely alignment is many times incorrect. Accuracy in sequence alignment can be improved significantly by the use of multiple sequence aligners, provided that several related

sequences are available for use. Today, at least for the purposes of identifying genomic variants at a single nucleotide scale, multiple alignment is the ”technology” of choice.

The main contribution of this section is a set of novel algorithms for identifying structural differences among multiple genomes through the use of multiple sequence comparison methods. Our algorithms will help better analyze vast amounts of publicly available genomic sequence data (e.g. 1000 genomes project [5, 124]), which include WGSS data from diverse populations (and members of the same population or even family).

Available methods for SV discovery typically employ paired-end sequencing: inserts from a donor genome (from a tightly controlled length distribution) are read at two ends, which are later aligned to a reference genome. Provided that the mapping loci is correctly identified, an increase or decrease of the distance between the end reads indicate an insertion or a deletion. We refer the reader to the Introduction section of this thesis for a summary of SV discovery methods, or the surveys [6, 105]

In the previous chapter we demonstrated, for example, that on the well known NGS genomes of the Yoruban family (involving a child, the mother and the father, NA18506, NA18507, NA18508) [21], the independent application of VariationHunter (the only publicly available algorithm for *Alu* discovery on NGS genomes) predicts up to 410 de novo *Alu* inserts in the child! A careful inspection of the clusters obtained by VariationHunter on all three individuals on the other hand, reveals that *all* of these 410 novel *Alu* inserts predicted are indeed *false positives* mostly due to single nucleic variation (SNVs) or varying read coverage, etc.

Note that soft clustering strategies for SV detection between one donor genome and a reference genome do provide both false positives, as well as false negatives, due to SNV effects and others. However, the proportion of false positives among all positives predicted will be low because of the high number of actual SVs typically observed between a donor and the reference. On the other hand, when the goal is to identify structural differences between two highly related donors, i.e. donor one by (D_1) and donor two (D_2), by using the reference (R) as an intermediary, while the number of false positives (between D_1 and R and D_2 and R) will be of similar scale, the proportion of false positives among all positives will be high, simply due to the low number of actual SVs that would be present between the donor genomes. Thus although VariationHunter (and other soft clustering strategies) may provide high levels of accuracy for SV detection between one donor and the reference genome, it may provide a low level of accuracy when finding the structural differences between two (or more) donor genomes.

The CommonLAW Approach. For the purpose of addressing the above issues arising in soft clustering techniques, we introduce the problem of *simultaneous* SV discovery among multiple paired-end NGS donor genomes - with the help of a complete reference genome. In order to solve this problem, we also introduce novel combinatorial algorithms, which we collectively call `CommonLAW` (Common Loci structural Alteration discovery Widgets). `CommonLAW` aims to predict SVs in several donor genomes by means of minimizing a *weighted sum of* structural differences between the donor genomes as well as one reference genome.[1] The (pairwise) weights are a function of (1) the expected genomic proximity of the individual donors sequenced (see details in the Results Section), and (2) type, loci and length of the individual structural alterations considered. The problem of minimizing (for example, sum-of-pairs) genomic alterations between multiple genomes is NP-hard. In this section we describe a tight (i.e., asymptotically the best possible) approximation algorithm for the general simultaneous SV discovery problem - this algorithm is at the heart of the `CommonLAW` package. In addition, `CommonLAW` includes several efficient algorithms and heuristics for some special cases of the problem.

We have tested `CommonLAW` on the genomes of three Yoruban (YRI) individuals (mother-father-child trio) sequenced by Illumina Genome Analyzer with about $30x$ coverage (i.e. 3.24×10^{11} bp of sequencing data), for the purpose of predicting deletions and *Alu* insertions. We compare the deletion predictions with the validated deletions reported in [5, 124]. We compare the *Alu* insertion predictions with *Alu* polymorphism loci reported in dbRIP [67]. In both cases we observe that `CommonLAW` provides a much higher level of accuracy in comparison to VariationHunter, the only publicly available computational method for *Alu* insertion discovery in NGS genomes.

In addition, we have tested `CommonLAW` on a high coverage parent-offspring trio of European ancestry from Utah (CEU), recently sequenced and analyzed by the 1000 Genomes Project [5]. We demonstrate the predictive power of `CommonLAW` by comparing its calls with the validated deletions reported in [5, 124].

[1] Although it is easy to generalize the formulation we provide here to multiple reference genomes, we do not explore this problem here due to the lack of alternative, completely - and independently assembled reference genomes.

4.1 Methods

4.1.1 Simultaneous Structural Variation Discovery among Multiple Genomes

Given a reference genome and a collection of paired-end sequenced genomes, $G_1, \cdots, G_\lambda$, Simultaneous Structural Variation Discovery among Multiple Genomes (SSV-MG) problem asks to simultaneously analyze the genomes so as to predict structural differences between them and the reference genome. For solving the SSV-MG problem, notice that a paired-end read from a genome G_k with no *concordant* alignment on the reference genome suggests an SV event in G_k [147, 149]. Unfortunately, if the number of *discordant* alignment locations of a paired-end read is more than one, the paired-end read potentially supports several SV events. The crucial question we try to answer in this section is among all potential SV events supported by a discordant paired-end read, which one is correct? In the presence of a single donor genome, one answer to this question was given by Hormozdiari et al. [55] with the introduction of novel approximation algorithms for the "Maximum Parsimony Structural Variation" (MPSV) discovery problem. In [55], a SV cluster is defined as a set of discordant (paired-end read) alignments that can support the same potential SV event; similarly, a *maximal* SV cluster is defined as an SV cluster to which no other alignments could be added [16, 136, 55, 57, 48]. A maximal SV cluster is considered to be a valid cluster if it satisfies a certain set of mathematical rules specifically defined for each SV event type. [55, 57, 48].

As defined in [55], the MPSV problem for a single donor genome asks to compute a unique *assignment* for each discordant paired-end read to a maximal valid SV cluster such that the total number of implied SVs is minimized. The SSV-MG problem, which generalizes the MPSV problem to multiple donor genomes, also asks to identify a set of maximal SV clusters to which each discordant paired-end read can uniquely be assigned - under the maximum parsimony criteria. A solution of the SSV-MG problem is said to provide support for each SV cluster as a function of the discordant paired-end reads it assigns to the SV cluster. Intuitively, if the support comes from paired-end reads from a large number of - especially highly related - genomes (e.g. members of a family), the SV event is more likely to be "correct". The maximum parsimony criteria we employ is formulated to reflect this observation as follows. Each SV event in a solution to the SSV-MG problem is associated with a weight, which is a function of the set of the donor genomes on which the SV event is present (i.e. has at least one discordant paired-end read mapping which is assigned to the associated SV cluster). If an SV event is present among many donor genomes, its weight will be relatively small; on the other hand a SV event which is unique to only one donor genome will have a larger weight. In this setting, the SSV-MG problem asks to identify a set of SV events whose

total weight is as small as possible.

4.1.2 The Algorithmic Formulation of the SSV-MG Problem

Given an NGS sequenced donor genome G_k, let the set of its discordant reads (i.e. the reads which do not have a concordant mapping) be $R^k = \{pe_1^k, pe_2^k, \cdots, pe_{n_k}^k\}$; thus n_k denotes the number of discordant reads of G_k. Let $n = \sum_{k=1}^{\lambda} n_k$ be the total number of discordant reads among all the donor genomes and let $R = R^1 \cup R^2 \cup \cdots \cup R^\lambda$ be the set of all discordant reads. For the algorithmic formulation of the SSV-MG problem, the donor genome G_k and all its discordant reads are said to be of "color" k.

Note that each discordant read may have several alignment locations on the reference genome so, as we discussed earlier, the aim is to find a unique assignment of each discordant read in R to exactly one of the maximal SV clusters (and, hence, to one potential SV event). (For detailed definitions of discordant reads and multiple paired-end read alignments, please see [55].)

Let $\mathcal{S}$ be the set of all maximal valid clusters. For each $r \in R$, let $\Psi_{\mathcal{S}}(r) \subseteq \mathcal{S}$, denote the set of all maximum valid clusters "supported by" r, i.e. for which r has an associated alignment. For each possible subset of colors (i.e. donor genomes) $\mathcal{C} \subseteq \{1, \cdots, \lambda\}$), we define a weight, $\omega_{\mathcal{C}}$, as a measure of genetic affinity between the donor genomes in this subset. E.g. the weight of a subset of two donor genomes can be defined as the estimated ratio of the total number of SVs in the two genomes and the number of shared SVs in the genomes - i.e. the likelihood of an SV event being shared among the two donor genomes, rather than being present in only one donor genome. Note that currently in the implementation of the algorithm, the values of the weights $\omega_{\mathcal{C}}$ are set in an ad-hoc manner, for related or unrelated genomes. We set the weights empirically and inversely proportional to the number of events we expect to observe in each subset of genomes. Then we can define the weight of an SV event (i.e. maximal valid cluster) s, denoted w_s, as $\omega_{\mathcal{C}(s)} \times \Delta_s$ where $\mathcal{C}(s)$ is the set of donor genomes sharing the SV event s and Δ_s is a measure of the likelihood of the SV event s, which depends only on the length and the type of s.

Based on these notions, Simultaneous Structural Variation discovery among Multiple Genomes (SSV-MG) problem asks to *assign* each discordant read $r \in R$ to one of the maximal valid SV clusters in $\Psi_{\mathcal{S}}(r)$ such that the following optimization function (`COST`) is minimized:

$$\texttt{COST} = \sum_{\forall s \in \mathcal{S}} I_s \cdot w_s = \sum_{\forall s \in \mathcal{S}} I_s \cdot \omega_{\mathcal{C}}(s) \cdot \Delta_s.$$

Here I_s is an indicator variable equal to one, if there if at least one discordant read assigned to s (i.e. s is selected); otherwise I_s is equal to zero.

SSV-MG for two donor genomes. A special case of the SSV-MG problem is on comparing two donor genomes by the use of a reference genome as an intermediary. The motivation for the study of this special case is two-fold; From a theoretical point, we can guarantee that our proposed algorithm is the best possible solution for this case. From a practical point, this case obviously applies to two highly related genomes such as those from healthy v.s. tumor tissues of an individual, for the purpose of identification of common and distinct SV events with respect to the reference genome, and, as a result, the structural differences between them.

We study this case through a combinatorial problem, namely *Red-Black-Assignment* problem where two colors, *red* and *black* are respectively associated with the two donor genomes (and their discordant paired-end reads). We call an SV event, which has assigned paired-end reads from both colors, *multicolor* and call an SV event which has assigned paired-end reads from a color red/black respectively *red* or *black*. Clearly a multicolor SV event indicates no structural difference between the two genomes whereas a red or a black SV event indicates a structural difference.

Let $\mathcal{M}$ be the set of multicolor SV events, $\mathcal{R}$ be the set of red events, and $\mathcal{B}$ be the set of black events. SSV-MG problem for this particular case asks to find a solution that minimizes the following cost function: $\texttt{cost} = \omega_{\{red,black\}} \sum_{s_m \in \mathcal{M}} \Delta_{s_m} + \omega_{\{black\}} \sum_{s_b \in \mathcal{B}} \Delta_{s_b} + \omega_{\{red\}} \sum_{s_r \in \mathcal{R}} \Delta_{s_r}$. Clearly a lower value of $\omega_{\{red,black\}}$ in comparison to $\omega_{\{red\}}$ or $\omega_{\{black\}}$ asks for a more conservative estimate of the structural differences between the two genomes.

In the next section, we show that the SSV-MG problem is NP-hard to solve exactly. In fact, it is also NP-hard to solve within an approximation factor of $c\frac{\omega_{\max}}{\omega_{\min}} \log n$ (for some constant c); this is the case even when $\Delta_s = 1$ for all SV events s. Note that n is the total number of discordant reads, $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum possible weight for of an SV event, respectively. (Intuitively, $\omega_{\min}$ is the weight of a multicolor SV event which has assigned reads from all different colors and $w_{\max}$ is the weight of an SV event which has only assigned reads from one specific color.)

4.1.3 Hardness of approximating the general SSV-MG problem

We use an approximation preserving reduction from the well known set cover problem. The set cover problem is defined as follows: Given a universe U with n elements and a family $\mathcal{S}$ of subsets of U (i.e $\mathcal{S} = \{S_1, \cdots, S_m\}$), we want to find the minimum number of sets in $\mathcal{S}$ whose union is U. Raz and Safra [120] proved that there exists a constant d such that the set cover problem cannot be approximated within $d \log n$ unless $P = NP$. Alon, Moshkovitz and Safra [110] showed the similar complexity result also holds with a smaller constant. We use this complexity result to prove

that the SSV-MG problem cannot be approximated within a constant times $\frac{\omega_{\max}}{\omega_{\min}} \log n$.

Lemma 1. *There exists a constant d such that the set cover problem cannot be approximated within $d \log n$ even in the case where the size of the optimal solution for the problem is already known.*

Proof. Given a set cover instance, we define OPT as the size of its optimal solution. Note that OPT is always an integer smaller than or equal to m, where m is the size of the family of subsets of $\mathcal{S}$. We show that if there exists a black-box which finds a solution with a size $d \log n \cdot OPT$ for the case where OPT is already known, the set cover problem can also be approximated within the same factor. This reduction would be in contradiction with the complexity result of Alon et al. [110]. Assume there exists such a black-box that finds an approximated solution with at most $d \log n \cdot OPT$ in polynomial time. For each integer $i \in \{1, \cdots, m\}$, we can now guess the value of OPT to be equal to i and execute m different black-boxes (i.e. for each i) in parallel. Next, we verify the outputs of those black-boxes terminated in polynomial time and find an approximated solution within the same factor for the general set cover problem. □

Theorem 2. *There exists a constant c such that SSV-MG has no approximation factor smaller than $\big(\lambda - 1 + \frac{\omega_{\max}}{\omega_{\min}} \cdot (c \log n - \lambda + 1)\big)$, unless $P = NP$.*

Proof. We use a reduction from the set cover problem where the size of its optimal solution is already known. For simplicity, we call this problem Set Cover Optimal Known (SC-OK) throughout this proof. Suppose we are given an SC-OK instance with $U = \{x_1, \cdots, x_n\}$ and $\mathcal{S} = \{S_1, \cdots, S_m\}$ as its universe and family of subsets, respectively, and let OPT be the size of its optimal solution. We construct an instance of the SSV-MG problem as follows: For each color $\ell (1 \leq \ell \leq \lambda - 1)$ and for each $j (1 \leq j \leq OPT)$, we introduce a new element $y_{\ell,j}$ with the color ℓ. The color of the elements in U is set to λ. Let $Y = \{y_{\ell,j} | 1 \leq \ell \leq \lambda - 1, 1 \leq j \leq OPT\}$ be the set of all these new elements. We define $U' = U \cup Y$, as a new universe for the instance of SSV-MG and construct its family of subsets $\mathcal{S}'$ as follows: Corresponding to each $S_i \in \mathcal{S}$, we have a subset $S'_i = S_i \cup Y$ in $\mathcal{S}'$. In other words, all the subsets in the family will share all of the new $(\lambda - 1)OPT$ elements. It can be seen that an optimal solution for SC-OK gives an optimal solution for SSV-MG with a cost equal to $\omega_{\min} \cdot OPT$ since all the selected subsets can have all different λ colors assigned to them. Furthermore, any feasible solution for SC-OK with $k \geq (\lambda - 1)OPT$ subsets gives a solution with the cost of at least $\omega_{\max} \cdot [k - (\lambda - 1)OPT] + \omega_{\min} \cdot (\lambda - 1)OPT$ for SSV-MG. We have $(\lambda - 1)OPT$ new elements with colors from 1 to $\lambda - 1$ (i.e. other than λ) and even if (1) we assign these new elements to $(\lambda - 1)OPT$ different subsets and (2) $\omega_{\min}$ is equal

to the weight of an SV event with assigned paired-end reads from two colors, the cost of SSV-MG cannot become less than $\omega_{\max} \cdot [k - (\lambda - 1)OPT] + \omega_{\min} \cdot (\lambda - 1)OPT$[2].

We claim that if there exists an algorithm which gives an approximate solution within a factor of $\lambda - 1 + \frac{\omega_{\max}}{\omega_{\min}} \cdot (c \log n - \lambda + 1)$ for the SSV-MG instance, then we can also give an approximate solution within a factor of $c \log n$ for SC-OK. As discussed earlier, the optimal solution for this SSV-MG instance has a cost $\omega_{\min} \cdot OPT$ and if the algorithm guarantees the desired factor, the cost of the solution would be at most $\omega_{\min} \cdot OPT \cdot \big(\lambda - 1 + \frac{\omega_{\max}}{\omega_{\min}} \cdot (c \log n - \lambda + 1)\big)$. Now we will show that, in this case, the total number of subsets in the solution returned will become less than $c \log n \cdot OPT$ which contradict with the result of Alon et al [110] for a small constant c.

Assuming k is the total number of subsets in the solution, we have:

$$(\lambda-1)OPT \cdot \omega_{\min} + (k - (\lambda-1)OPT) \cdot \omega_{\max} \leq \omega_{\min} \cdot OPT \cdot \big(\lambda - 1 + \frac{\omega_{\max}}{\omega_{\min}} \cdot (c \log n - \lambda + 1)\big)$$

Thus,

$$\lambda - 1 + \frac{k - (\lambda-1)OPT}{OPT} \cdot \frac{\omega_{\max}}{\omega_{\min}} \leq \lambda - 1 + (c \log n - \lambda + 1)\frac{\omega_{\max}}{\omega_{\min}} \Rightarrow \frac{k - (\lambda-1)OPT}{OPT} \leq (c \log n - \lambda + 1)$$

Thus,

$$k \leq (c \log n \cdot OPT)$$

So these k subsets will give a feasible solution within $c \log n$ to SC-OK which contradicts the complexity result of Alon et al. [110], for a sufficiently small constant c.

□

4.1.4 A simple approximation algorithm for SSV-MG problem

It is possible to obtain an approximate solution to the SSV-MG problem within an approximation factor matching the lower bound mentioned above, when $\Delta_s = 1$ for all SV events s - in near-linear time. For that we adopt the greedy algorithm for approximating the well known set cover problem [148] to obtain a solution within $O(\log n \cdot (\omega_{\max}/\omega_{\min}))$ factor of the optimal solution for the SSV-MG problem. (Again, $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum possible weights among all SV clusters.) The resulting algorithm, which we call *Simultaneous Set Cover* method (*SSV*), selects sets iteratively: at each iteration, it selects the set which contains the largest number of elements not

[2]Note that, since $\omega_{\min}$ is usually much smaller than the weight of events with two assigned colors and $\omega_{\max}$, we will get a much better bound in reality.

previously covered. For a given instance of the SSV-MG problem and its corresponding set cover instance, we denote the respective sizes of their optimal solution by OPT_{SSV} and OPT_{SC}. It is easy to see that $OPT_{SSV} \geq \omega_{\min} \cdot OPT_{SC}$, since at least OPT_{SC} subsets have to be selected in SSV-MG to cover all the elements of the universe and each of those subsets have a weight of at least $\omega_{\min}$. The greedy solution for the set cover problem gives a solution of at most $\log n \cdot OPT_{SC}$, hence, the same solution for SSV-MG will have a cost of at most $\omega_{\max} \log n \cdot OPT_{SC}$. Thus, the SSC method gives an $O(\log n \cdot (\omega_{\max}/\omega_{\min}))$ approximation for the SSV-MG problem - which matches the lower bound indicated for the SSV-MG problem above.

4.1.5 A maximum flow-based update for Red-Black Assignment

Although *SSV* method described above is very fast, the results it provides could be far from optimal. However it is possible to improve the results of the *SSV* method through an additional (post processing) step as follows. The *SSV* method picks a collection of valid clusters such that each discordant read is assigned to exactly one cluster. Each cluster is either multicolor or is considered Red or Black depending on whether it contains reads from both colors (i.e. donor genomes) or from a single color. The additional step we describe here does not change the clusters and the SV events they support. Rather, assuming that the clusters are "correct", the additional step reassigns the discordant reads to the clusters so as to maximize the number of multicolored clusters. As a result of this assignment, we may end up with empty clusters; we simply discard these clusters and return the non-empty clusters as an (improved) solution to the SSV-MG problem. Note that the additional step is guaranteed to return a solution which is at least as good as the one returned by the *SSV* method; in many cases the solution will be much better. Unfortunately, it can only be applied to the SSV-MG problem when the number of colors (i.e. donor genomes) is exactly two. Even for three colors, the problem of maximizing multi-colored clusters (i.e. those clusters with reads coming from all three donor genomes) is NP-hard (this is one of the first 21 NP-complete problems discovered by Karp [71])

The additional step formulates the re-assignment problem (of discordant reads to clusters) as a maximum flow problem as follows. Consider an instance of the Red-Black-Assignment problem and let $S_{SELECTED} = \{S_1, \cdots, S_k\}$ be the subsets of the family $\mathcal{S}$ which are already selected in a solution. Let $R = \{r_1, \cdots, r_{n_R}\}$ be the set of red elements and $B = \{b_1, \cdots, b_{n_B}\}$ be the set of black elements, where $n_R + n_B = n$ (i.e. the number of elements in the universe). We construct a network $\mathcal{G}$ as follows: for $1 \leq i \leq k$, each S_i is represented by a vertex in the network and

corresponding to every element in the universe, we have a vertex in the network in $\mathcal{G}$. For every pair (r_i, S_p) such that r_i is a member of S_p, we have an edge with a capacity equal to one and for every pair (S_q, b_j) such that b_j is a member of S_q, we have an edge (S_q, b_j) with a capacity one. A source vertex `SOURCE` is connected to all vertexes in R and all vertexes in B are connected to a sink vertex `SINK`. All the internal vertices (i.e. all vertices except the sink and the source) have capacity one as well.

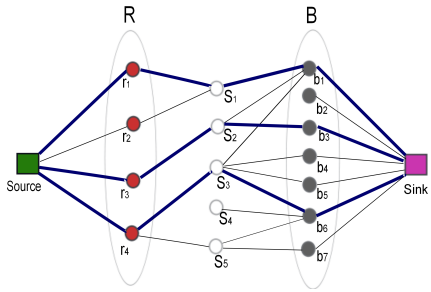


Figure 4.1: The set $R = \{r_1, r_2, r_3, r_4\}$ represents the red elements and $B = \{b_1, b_2, \cdots, b_7\}$ represents the black elements. The selected subsets are $S_1 = \{r_1, r_2, b_1\}, S_2 = \{r_3, b_1, b_3\}, S_3 = \{r_4, b_1, b_4, b_5, b_6\}, S_4 = \{b_6\}, S_5 = \{r_4, b_6, b_7\}$. All the edges and vertices have capacity one and the maximum flow is shown in dark blue. As it can be seen here, the maximum flow solution re-assigns the reads so that three sets/clusters, $s1$, $s2$, $s3$, become multicolor and one set/cluster, $s4$ becomes empty - thus its associated potential SV event will not be among the predicted SV events by our method.

Our additional postprocessing step computes the maximum (integral) flow from s to t, and identifies all edges (r_i, S_ℓ) and (S_ℓ, b_j) in $\mathcal{G}$ with unit flow in the network, and *re-assigns* the elements r'_i and b'_j to the subset S_ℓ. Observe that a solution to this maximum flow problem will maximize the number of multi-color subsets. Figure 4.1.5 demonstrates an example of how the network is constructed and how a solution to the maximum flow problem re-assigns the discordant reads to clusters.

4.1.6 An $O(1 + \frac{\omega_{\max}}{\omega_{\min}})$ approximation algorithm for limited read mapping loci

It is possible to further improve the algorithms presented above for the special case where each discordant read maps to a *small* number of loci on the genome. For simplicity, we present the limited case that each discordant read maps to *exactly* two locations - i.e., in Red-Black assignment problem terms, each element is a member of exactly two subsets. The generalization of this case to a more general one, where each read can be present in at most f clusters is not very difficult and is omitted.[3]

[3]E.g. for the generalization to the case where each element is in *at most* two subsets, observe that if an element is in only one subset, that subset must be included in any feasible solution.

The special case, which we denote as the Red-Black-Assignment-F2 problem, has a graph theoretical formulation similar to the vertex-cover problem. Let G be a simple graph for which there is a vertex s_i corresponding to each subset S_i in the family $\mathcal{S}$ and there is an edge $e = (s_i, s_j)$ corresponding to each element e in U - provided e is in both S_i and S_j. The edges of G are labeled with the color of their corresponding elements (either red or black). In order to solve the Red-Black-Assignment-F2 problem, all we need to do is to set an orientation to each edge: the vertex (corresponding to a cluster) for which a given edge (corresponding to a read) is pointing to gives the cluster to which a read is assigned. The Red-Black-Assignment-F2 problem thus reduces to setting an orientation to the edges in this graph such that $\alpha \cdot \omega_{\min} + \beta \cdot \omega_{\max}$ is minimized: here α is the number of vertices to which edges of both colors are pointing and β is the number of vertices to which edges of only one color are pointing.[4]

Let OPT be the minimum number of vertices required to cover all the edges (i.e. the size of a minimum vertex cover). It is easy to see that $\omega_{\min} \cdot OPT$ is a lower bound for Red-Black-Assignment-F2 and the simple greedy algorithm of the vertex cover problem [148] gives a $2 \cdot \frac{\omega_{\max}}{\omega_{\min}}$ approximation.[5] The following algorithm achieves a smaller (in fact, the best possible) approximation factor.

Denote the instance of orientation setting problem (to which the Red-Black-Assignment-F2 problem is reduced) by H. It is possible to compute a *maximal matching* (of vertices) in this graph in polynomial time; let M denote this matching. Suppose M has p edges with red labels and q edges with black labels where $p + q = |M|$. Let $R = \{r_1, r_2, \cdots, r_p\}$ be the set of red edges and $B = \{b_1, b_2, \cdots, b_q\}$ be the set of black edges in the maximal matching.

(1) Consider an edge e_M in this matching and suppose that the optimal solution to the orientation setting problem points $e_M \in M$ to a multicolor vertex; also suppose that the other vertex e_M is incident to is not pointed by any other edge. Thus, in the optimal solution, the "cost" of covering each of the edges e_M in the maximal matching m is at least $\omega_{\min}$. Our algorithm covers each such edge e_M by selecting both vertices it is incident to, incurring a cost of $\omega_{\max} + \omega_{\min}$.

(2) If the optimal solution covers e_M with a unicolor vertex it is incident to, our algorithms covers it with a cost of at most $2 \cdot \omega_{\max}$, again by picking both vertices.

Provided the two objectives above are achieved, our algorithm guarantees an approximation factor of $1 + \frac{\omega_{\max}}{\omega_{\min}}$.

---

[4] without loss of generality, we assume that $\omega_{\{red\}} = \omega_{\{black\}} = \omega_{\max}$.

[5] The greedy algorithm selects at most $2OPT$ unicolor subsets.

Theorem 3. *Red-Black-Assignment-F2 problem can be approximated within a factor of* $1 + \frac{\omega_{\max}}{\omega_{\min}}$.

Proof. Our algorithm first probes all the edges in R (the set of red edges in the maximal matching) and assigns them to one of their vertices. Each red edge $r_i \in R$ is from one of the following categories:

- *There exists a black edge specific to r_i in H:* in other words, this black edge shares a vertex with r_i but does not share a vertex with any other red edge in R. In this case, the algorithm simply orients both r_i and the above-mentioned black edge to this shared vertex.

- *r_i does not share a vertex with a black edge in H:* In this case the algorithm orients r_i arbitrarily.

- *Each black edge sharing a vertex with r_i has its other vertex shared by another red edge $r_j \in R$:* Let $R' \subseteq R$ be the set of red edges which share a vertex with a black edge - not specific to any red edge. We construct a *new* graph $H^{R'}$ as follows: corresponding to each edge $r'_j = (x'_j, y'_j)$ in R' set up a *vertex* ρ'_j in $H^{R'}$. For each pair of vertices ρ'_k and ρ'_ℓ in $H^{R'}$ and for each black edge in H which share vertices with both r'_k and r'_ℓ, set up an edge $e'_{k,\ell}$ connecting ρ'_k and ρ'_ℓ. Note that $H^{R'}$ is not necessarily a simple graph. Suppose $H^{R'}$ has t connected components denoted by $C_1, \cdots, C_t$. For each C_i, we first orient its edges such that each vertex has an *indegree* at least 1. Note that such an orientation can always be discovered via a Depth-first search (DFS) algorithm, unless C_i is a (simple) tree in which exactly one vertex (the root of the DFS) would have indegree equal to zero (i.e. no edges terminating at it). WLOG, let the direction of the edge $e'_{k,\ell}$ be from ρ'_k to ρ'_ℓ. We orient the black edge $e'_{k,\ell}$ towards its vertex (say x_ℓ), which is shared by r'_ℓ. The edge r'_ℓ will also be oriented to x_ℓ and thus x_ℓ will be multicolor. This guarantees that all but one of the red edges in R' will be oriented towards a vertex, also oriented by a black edge.

We will use a similar strategy for the set of black edges in the matching and finally orient all the remaining edges in H arbitrarily. This strategy will guarantee that even if the optimal solution covers an edge $e_M \in M$ with a multicolor vertex and does not pick the other vertex of e_M (i.e. incurring a cost of only $\omega_{\min}$), e_M can be covered with a cost of at most $\omega_{\max} + \omega_{\min}$ by selecting both of its vertices - which will ensure at least one of its vertices will be multicolor. If the optimal solution covers e_M with a single colored vertex, our strategy will cover it with a cost of at most $2 \cdot \omega_{\max}$, providing us a $1 + \omega_{\max}/\omega_{\min}$ approximation factor.

□

4.1.7 Efficient heuristic methods for the SSV-MG problem

In addition to the approximation algorithms given above, we provide two heuristics for solving the SSV-MG problem efficiently. The first heuristic uses the weights ω_s to calculate a *cost-effectiveness* value for each cluster s, while the second heuristic deploys the concept of *conflict resolution* (introduced in [57]) to obtain more accurate results in diploid genomes.

Simultaneous Set Cover with Weights (SSC-W). The first heuristic is a greedy method similar to the weighted set cover algorithm [148] with one major difference. Here the weight w_s of each subset s is not fixed throughout the algorithm, but rather is dependent on the elements which are assigned to that subset - more precisely the weight is a function of how closely related the colors (i.e. donor genomes) assigned to that subset are. Because during the execution of the method, the colors assigned to each subset can change, so can the weight of that subset.

The method selects the SV clusters in an iterative greedy manner based on their "cost-effectiveness" value in each iteration. In a given iteration, the method selects the set with the highest "cost-effectiveness" value, based on the maximum number of colors that can be assigned to the set in that iteration. The cost-effectiveness of a SV cluster s in the i iteration is equal to $\frac{w_{s_i}}{|s_i|}$, where w_{s_i} is the weight of the subset of s which is not yet covered (i.e. the reads in s which are not covered till the i iteration). Note that this greedy method will guarantee an approximation factor of $O(\frac{\omega_{\max}}{\omega_{\min}} \log n)$.

Simultaneous Set Cover with Weights and Conflict Resolution (SSC-W-CR) The second heuristic employs the concept of *Conflict Resolution* and takes the diploid nature of the human genome into consideration. Hormozdiari et al. [57] introduced a set of mathematical rules to prevent selecting SV events that cannot be happening simultaneously in reality in a haploid genome.[6] The Conflict Resolution feature of this heuristic is based on those rules. Note that in [57] we have modeled the conflicting SV events in a "conflict graph" where each cluster is represented by a vertex. Two vertices are connected with an edge if the two SVs implied by the clusters are in conflict (please see A for a detailed case study). In SV detection in diploid genomes, a conflict free set of SVs is equivalent to a set of vertices that do not create a "triangle" in the conflict graph. In this heuristic we extend the above notion from a single genome to multiple genomes, such that we are not allowed to assign the same color to three clusters (vertices) forming a triangle in the conflict graph. We have devised an iterative greedy method which selects clusters based on their cost-effectiveness: the

[6] E.g. two clusters which indicate a deletion and significantly overlap with each other are considered to be conflicting.

cost-effectiveness of SV cluster s in iteration i is $\frac{w_{s'_i}}{|s'_i|}$, where s'_i is the subset of paired-end reads in s which are not covered until this iteration and do not conflict (i.e. create a triangle) with previously selected SV clusters that have a common color. More formally suppose that given the conflict graph $\mathcal{G}$, for each of the sets picked prior to iteration i, a subset of λ colors have been assigned to them. Any paired-end read $r \in s$ is considered to be a member of s'_i if:

- r is not covered by any of the $i-1$ clusters picked prior to iteration i,
- there is no pair of clusters q and p that have been picked in earlier iterations, such that q, p and r form a triangle and both q and p include the color of r.

The `CommonLAW` package is currently available at `http://compbio.cs.sfu.ca/strvar.htm` .

4.2 Experimental Results

We investigated the structural variation content of six human genomes in order to establish the benefits of Simultaneous Structural Variation discovery among Multiple Genomes (SSV-MG) compared to the Independent Structural Variation Discovery and Merging (ISV&M) strategy. The two data sets we investigate each constitutes a father-mother-child trio. The first trio is a Yoruba family living in Ibadan, Nigeria (YRI: NA18506, NA18507, NA18508 [21]). The second trio is a family from Utah with European ancestry (CEU: NA12878, NA12891, NA12892) sequenced with high coverage by the 1000 Genomes Project [5]. We aligned the downloaded paired-end reads to the human reference genome (NCBI Build 36) using *mrFAST* [7]. Statistics for each dataset are provided in table 4.1 (after removing low-quality paired-end reads).

Table 4.1: Summary of the analyzed human genomes

Individual	Population	# Reads	Read Length	Average Ins. size	Sequence Cov.	Physical Cov.
NA18506	YRI	3.444×10^9	35bp	222bp	40.1×	255×
NA18507	YRI	2.261×10^9	36-41bp	208bp	27.1×	157×
NA18508	YRI	3.175×10^9	35bp	203bp	37×	214×
NA12878	CEU	1.049×10^9	36-76bp	201bp	32.3×	70×
NA12892	CEU	0.510×10^9	35-51bp	153bp	12.6×	26×
NA12891	CEU	0.551×10^9	35-51bp	148bp	13.8×	27×

NA18506, NA18507 and NA18508 are the YRI child, father and mother respectively.
NA12878, NA12891 and NA12892 are the CEU child, father and mother respectively.

We sought to establish whether simultaneous analysis of all 3 genomes (in each trio) would result in more accurate detection of structural variation events in comparison to the conventional two step approach of ISV&M. For each of the trios, we analyzed the three genomes independently using the ISV&M based approach and simultaneously using the SSV-MG framework; we then compared the results from each analysis.

For the ISV&M approach we proceeded as follows:

- **The ISV step:** We analyzed each genome independently, using VariationHunter [55] for deletions and extended VariationHunter for *Alu* insertions [57]. [7].
- **The M step:** To identify common structural variation among different genomes, we compared each data set and merged shared structural variation predictions. Two structural variation predictions were considered to be "shared" (i.e. they are the same variation in two different individuals) if the ends of each selected cluster were within 200bp from each other. Finally, the support value of each shared SV is considered to be the total paired-end reads in the two (or more) individuals which support that shared SV.

In these experiments we were purely interested in evaluating the added benefit of simultaneous analysis over independent analysis. We used VariationHunter [55], a maximum parsimony based approach, for the ISV&M analysis since all the SSV-MG methods proposed here are also maximum parsimony based methods. In addition VariationHunter is one the very few tools with capability to find transposon insertions.

The experiments in this section focus on two types of structural variations:

- Transposon insertions (i.e. *Alu* insertions) on YRI data set
- Medium and large-size deletions on YRI and CEU data sets

4.2.1 *Alu* element insertions

As we have described earlier [57, 56], it is possible to use VariationHunter within the ISV&M framework, for discovering transposon insertions such as *Alu* elements on a WGSS donor genome with respect to a reference genome. Below we compare this approach, as a representative of the ISV&M framework, with the SSV-MG framework, more specifically SSC and SSC-W approximation algorithms.

[7] Note that any other method such as BreakDancer, MoDil or GASV could have been used in this step

We applied the ISV&M and the two SSV-MG approaches, namely SSC and SSC-W algorithms (we set two potential *Alu* inserts that overlap highly as the same *Alu*; this implies that conflict resolution is not necessary for this case) to the discovery of *Alu* insertions in the YRI trio. The results from each analysis were then compared to *Alu* polymorphism loci reported in dbRIP[67] - which provides an estimate on the tradeoff between the number of predictions made and the fraction of the known *Alu* insertions captured Since the contents of dbRIP are curated from a variety of data sources, for a given an *Alu* insertion prediction, a match in dbRIP is a good indicator that the prediction is a true positive. (Figure 4.2 provides an indirect comparative measure for the true positive/false negatives rate as a function of the calls made.) Note that we call a predicted *Alu* insertion, a match to an *Alu* insertion locus reported in dbRIP [67], if the reported locus in dbRIP is within 100bp of the breakpoints predicted.

For each method, we calculated the number of *Alu* insertions with a dbRIP match for a range of thresholds for the read support for each *Alu* insertion. The fraction of dbRIP matching *Alu* insertions is consistently higher for SSC and SSC-W methods in comparison to that of the ISV&M framework for all threshold values for support; see Figure 4.2.

Since the two data sets we considered each involve members of a family, it is expected that many of the *Alu* insertions observed are common to all three genomes. However, the ISV&M framework predicted only 507 common *Alu* insertion loci common to all three individuals, in contrast to 1044 common inserts predicted by the SSC method and 1257 common inserts predicted by the SSC-W method (See figures 4.3(a), 4.3(b), and 4.3(c)).

The rate of de novo *Alu* insertions is estimated to be one new *Alu* insertion per 20 births [33]. Thus, it is quite unlikely that the genome of the child (NA18506) in the YRI trio contains several *Alu* insertions that are not present in the parent genomes. However, the ISV&M framework based on VariationHunter [57], reported that among the top 3000 predicted loci [8], 410 were de novo (that is, unique to the child). This number clearly is extremely high given our current knowledge of *Alu* insertion rate. Thus the majority of these 410 events are likely to be misclassified as de novo events by the ISV&M framework. Interestingly, using the SSC algorithm, this number was reduced to only 20 de novo events among the top 3000 predictions (Figure 4.3(b)). Furthermore, the SSC-W algorithm [9] reduces the number of de novo *Alu* insertions to *zero* in the top 3000 *Alu* insertion predictions (see Figure 4.3(c)).

[8] The 3000 loci with the highest number of paired-end read support

[9] the weights used for SSC-W were derived from the fraction of *Alu* insertion common between the individuals reported by the SSC results

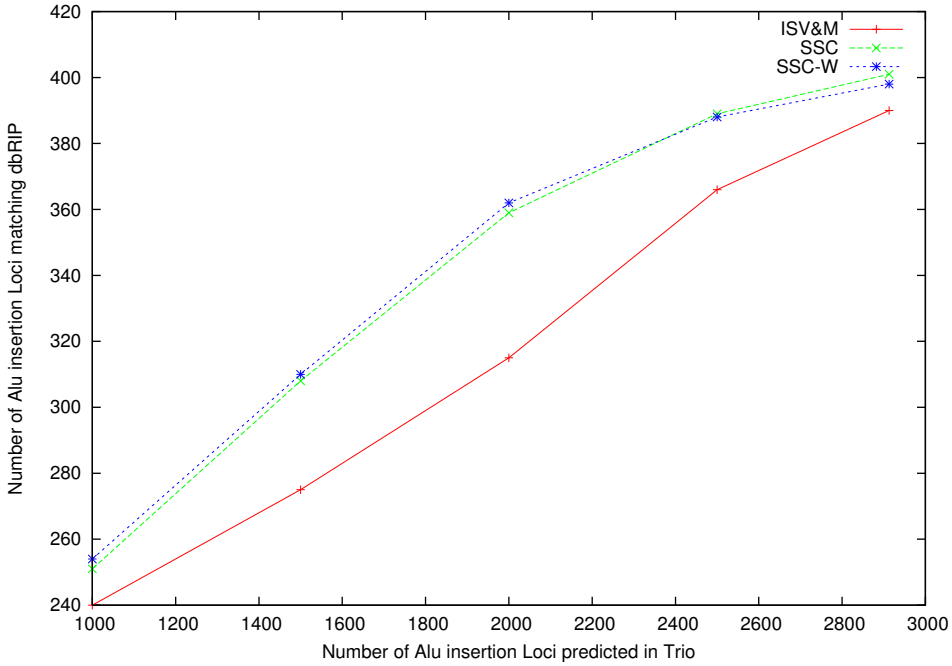
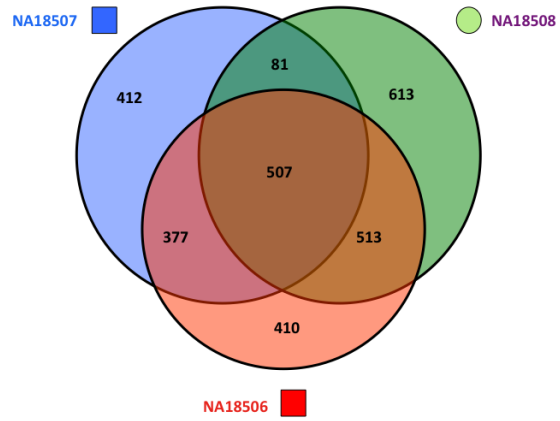


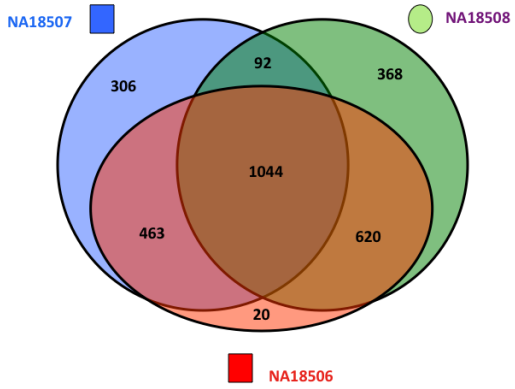
Figure 4.2: *Alu* insertion loci prediction and comparison with dbRIP: this figure shows the comparison of the *Alu* predictions made by the ISV&M, SSC and SSC-W algorithms which match *Alu* insertion loci reported in dbRIP (true positive control set). The x-axis represents the number of *Alu* insertions (with the highest support), while the y-axis represents the number of these insertions which have a match in dbRIP.

Note that one of the *Alu* insertion loci predicted as a de novo insertion in NA18506 by both the SSC method and the ISV&M framework turned out to be a locus experimentally tested positive for an *Alu* insertion by a Polymerase chain reaction (PCR) in the YRI trio ([56]). The result of PCR indicates that there is indeed an *Alu* insertion in the above locus in NA18506. However, it turned out that the insertion is not de novo but rather a transmission from the father (NA18507) to the child (NA18506). SSC-W, on the other hand, was able to correctly identify the *Alu* insertion in both NA18506 and NA18507 and thus was able to correctly classify the prediction as a transmitted event.

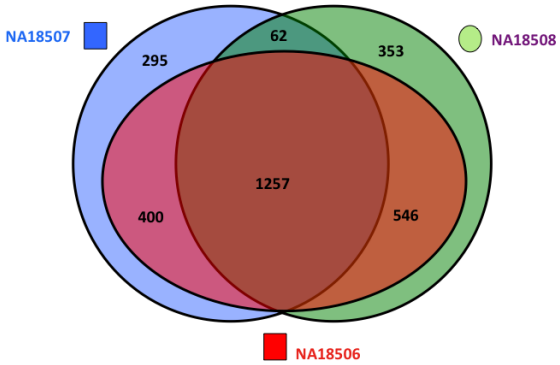
Note that a similar analysis on the CEU trio also revealed similar results to those we obtained on the YRI trio; please see [58] for details.



(a) ISV&M



(b) SSC



(c) SSC-W

Figure 4.3: (a), (b), and (c) detail the number of common and de novo events in each genome for ISV&M, SSC and SSC-W methods, respectively for the YRI trio (the top 3000 predictions were considered).

4.2.2 Deletions

In this section we compare the deletion calls made by algorithms proposed within the SSV-MG framework (i.e. SSC-W and SSC-W-CR) in comparison to those made by the ISV&M framework (i.e. VariationHunter [55]). The deletions considered here are medium-to-large scale events ($>$ 100bp and $< 1M$bp) in both YRI and CEU trios. In order to verify (at least some of) the predictions

made by the above algorithms, we used the validated SV events reported in the recent study by the 1000 Genomes Project Consortium ([124]). The results of this comparison are shown in table 4.2. In order to make the comparison as thorough as possible, we considered various sized subsets of calls for each method (obtained by varying the read support threshold on the predictions considered for each method) 9. In comparison to the ISV&M framework, the SSV-MG algorithms consistently produce a higher fraction of validated predictions in both YRI and CEU trios (see table 4.2)

We also compared the deletion predictions made by each one of the methods considered here on each individual genome from the CEU trio, with the validated deletion calls on the same individual genome by the above mentioned 1000 Genomes study ([124]). (Unfortunately such a set of validated deletions does not exist for the YRI trio.) Table 4.3 provides the number of the validated deletion calls from each specific genome in the CEU trio among the best supported 5000 calls made by each one of the methods considered here. Note that the number of de novo deletions reported in the child genome of CEU trio (NA12878) should not be high - as per the *Alu* insertions - as each deletion is likely to have been inhereted from one of the parents. Among the top 5000 deletion loci (on the child genome) predicted by the ISV&M framework, 84 were predicted to be de novo events. In contrast, among the top 5000 deletion loci (on the child) reported by the SSC-W algorithm, only 39 were predicted to be de novo events.

This reduction of more than 50% on number of misclassified deletions as de novo events demonstrates once again the improved predictive power of the SSV-MG framework over the ISV&M framework.

Table 4.2: Comparison of deletions discovered in CEU and YRI trio against validated deletions

Number of Predictions	CEU (NA12878, NA12891, NA12892)			YRI (NA18506, NA18507, NA18508)		
	ISV&M	SSC-W	SSC-W-CR	ISV&M	SSC-W	SSC-W-CR
2000	*728* (725)	*755* (751)	*1412* (1396)	*1280* (1279)	*1293* (1291)	*1536* (1520)
3000	*1058* (1058)	*1106* (1106)	*1780* (1763)	*1794* (1789)	*1797* (1794)	*2098* (2082)
4000	*1277* (1281)	*1342* (1345)	*2003* (1982)	*2192* (2183)	*2200* (2197)	*2554* (2534)
5000	*1449* (1457)	*1517* (1527)	*2139* (2121)	*2518* (2508)	*2537* (2534)	*2920* (2900)
6000	*1584* (1596)	*1667* (1678)	*2234* (2219)	*2771* (2765)	*2804* (2802)	*3207* (3186)
7000	*1659* (1674)	*1775* (1796)	*2314* (2305)	*2997* (2996)	*3040* (3042)	*3453* (3446)
8000	*1738* (1757)	*1861* (1886)	*2368* (2363)	*3192* (3195)	*3231* (3241)	*3662* (3682)
9000	*1797* (1816)	*1933* (1962)	*2398* (2396)	*3382* (3388)	*3417* (3434)	*3830* (3887)
10000	*1852* (1875)	*2005* (2038)	*2411* (2410)	*3512* (3532)	*3548* (3594)	*3970* (4084)
11000	*1892* (1918)	*2064* (2099)	*2420* (2422)	*3651* (3687)	*3694* (3757)	*4084* (4270)
12000	*1942* (1968)	*2118* (2159)	*2437* (2441)	*3753* (3787)	*3786* (3874)	*4173* (4425)
13000	*1960* (1988)	*2151* (2195)	*2445* (2457)	*3851* (3907)	*3887* (4003)	*4247* (4602)
14000	*1986* (2015)	*2177* (2225)	*2455* (2460)	*3958* (4010)	*3968* (4126)	*4314* (4756)

Deletions discovered for YRI and CEU trios (by 3 different approaches of ISV&M, SSC-W and SSC-W-CR) were compared against deletions reported by 1000 genomes project [124]. The loci of a reported deletion should be in the range of 300bp from a loci reported in [124] to be consider a "match". Different thresholds on number of predictions were considered for each method ranging from 2000 to 14000 (predictions by each method were sorted based on their support, and the top set of predictions were picked for comparison). The numbers given in *italic* font represent number of deletions reported in [124] (form the high coverage set) which match calls found by our methods, while the number in parenthesis represents number of deletions predicted by our methods (ISV&M, SSC-W and SSC-W-CR) which match reported deletions in [124].

Table 4.3: NA12878, NA12891 and NA12892 deletions discovered

Individuals	ISV&M	SSC-W	SSC-W-CR
NA12878	1349	1408	1723
NA12891	1191	1236	1468
NA12892	1351	1402	1814

The distribution of the validated deletion calls [124] among individual genomes in the CEU trio, specifically for the best supported 5000 predictions, by the three approaches.

Chapter 5

Identifying pairs of interacting protein partners

5.1 Introduction

The vast majority of cellular functions are exerted by combinations of interacting gene products. As a result, ”preservation of functionality” among proteins and other gene products typically implies ”preservation of interactions” across species. It is well established that protein-protein interactions (both physical interactions as well as co-occurence of domains) are preserved through speciation events (see [94, 113] and the references therein). A major implication of this is that the evolutionary trees behind two interacting protein families can look near-identical.

As interacting proteins have a tendency to co-evolve, it may be possible to assess the potential of two or more proteins (or other gene products) being interaction partners by measuring how similarly they evolve across related species. For this purpose a number of computational strategies have been developed. Such strategies aim to compare the phylogenetic trees of two (or more) protein or protein-domain families, where paralogs and orthologs are represented with leaves with appropriate labels and internal vertices can be interpreted as either speciation or duplication events. Among these strategies we will focus on *mirrortree* approaches, which explicitly or implicitly map leaves of a pair of trees (belonging to two distinct proteins or gene products) onto one another such that the leaves that are mapped to each other would be identified as potential interaction partners. Mirrortree approaches aim at an overall quantification of ”family similarity” via a measure of tree similarity. Typically these approaches do not aim to modify the specific topology of the underlying

phylogenetic trees and thus are different from tree reconciliation approaches [150]. They are also distinct from phylogenetic profiling methods [114], which aim to measure the phylogenetic profiles of proteins or domains to check for potential interaction partners.

The first mirrortree approach was proposed to discover protein-protein (rather than domain-domain) interactions and was based on comparing the distance matrices [1] resulting from the multiple alignment members of each protein family [112]. Note that one can interpret this as mapping leaves onto one another, as will be explained below. Since this study, a number of mirrortree approaches have been developed; almost all of these approaches are again based on comparing distance matrices rather than the trees directly. (See the introductory paper by [112] and [113] for more references.) In fact, direct comparison of gene trees has been considered as "... a problem yet to be fully resolved." [66, p. 2].

We consider a fresh approach to the problem of predicting protein or other gene product interactions by comparing gene trees directly, without the aid of a distance matrix. Note that such a distance matrix is a byproduct of the underlying phylogenetic tree: popular multiple sequence alignment methods typically align sequences in the order imposed by their phylogenetic tree and the "distances" in the matrix correspond to the distances in the phylogenetic tree. As a result our method should be considered as a more direct approach to mirroring trees.

In the case where there are no paralogs of any gene, assessing tree similarity is both computationally straightforward and reliable [113]. More specifically, if there is at most one family member per species, the mapping problem reduces to the problem of finding out species where the interaction is lost; after removal of such species, the topologies of the two trees will be identical i.e. the leaf representing a particular species in one tree will correspond to the leaf representing the same species in the other tree.

In the presence of paralogous genes (and thus proteins or gene products), however, the mapping problem becomes much more complex. For example, if we have n paralogs per tree for one of the species, we may need to evaluate more than $n!$ many potential mappings. ($n!$ is the number of mappings where each paralog from one tree pairs with a paralog from the other tree. In addition, there are mappings where one has to remove non-interacting paralogs. As was pointed out in [143], protein interaction can be preserved during duplication, while interaction can be lost during speciation.) Thus, the number of potential mappings is super-exponential in the number of paralogs per species, implying a significant computational challenge.

[1] The distance matrix of a gene tree is comprised of entries (i, j) which represent the distance between leaves i and j.

There are a number of mirrortree approaches which address the presence of paralogs and aim directly at inferring the correct mapping of leaves; these approaches typically aim to ”align” the distance matrices by shuffling and eliminating the rows (and corresponding columns) so as to maximize the similarity between the matrices. The similarity between two aligned matrices is defined in the form of root mean square difference [121], correlation coefficient [41], information-theoretic 'total interdependency' of multiple alignments [143], Student's t [66] or the size of the largest common submatrix [144]. Because an exact solution to the matrix alignment problem, where the goal is to maximize any of these notions of similarity (by determining the right mapping of rows and columns), is hard to compute, many available approaches employ heuristics based on swapping pairs of rows/columns in a greedy fashion. These methods also commonly perform column/row elimination from the ”larger” matrix only, and not the other [41, 66, 68, 121, 143]. We are aware of one exception by [144], which aims to determine the largest common (i.e. within a threshold) submatrix and removes the remainder of the columns and rows from both matrices. Similarly the only approach which directly compares the tree topologies themselves is by [68], which uses a Metropolis algorithm to heuristically travel 'tree automorphism' space. However, this approach cannot handle trees of different sizes. See [94, 113, 144] for references on mirrortree approaches which do not necessarily relate to the mapping problem in the presence of paralogs.

In this chapter we present polynomial-time algorithms that determine mappings of leaves which respect the topologies of the two trees compared. As input, we are given two ”gene trees” T and T' of two protein/domain families known to interact with one another. T and T' have labeled leaves where labels reflect species such that the presence of the same label at two different leaves reflects the presence of paralogs. We introduce and formally define the *gene tree alignment problem*, which aims to delete both leaves and inner vertices from both trees until the remaining trees are isomorphic, that is, one can map the vertices of the two remaining trees in a one-to-one fashion onto another such that ancestor relationships are preserved. This in particular implies a one-to-one mapping of the remaining leaves, which we present as output. Clearly, there are many different possible choices of such one-to-one mappings of leaves—our algorithms determine the *score-optimal* such alignment where different deletion operations are penalized in different ways, depending on how they transform the topologies of the trees. Note that our algorithm depends on some (user-defined) cost parameters, that can be used to impose constraints on the alignment. We describe the nature of our scoring scheme in detail in the following; please see the Methods section for full details and precise notations.

Note that the algorithm only outputs one uniquely determined, score-optimal alignment of subsets of leaves of T, T'. Note further that we do not perform an exhaustive search since we never consider mappings of leaves which imply mappings of internal vertices that do not preserve ancestor relationships of the gene trees T, T' and thereby contradicts their topologies.

Our method can be viewed as an extension of tree-edit distance approaches. Alternative constraints leading to polynomial time solvable variants on the tree edit distance is surveyed in [161]. For further, more recent work see also [116] that address the subtree homeomorphism problem, which, given a "text" tree T and a "pattern tree" P as the input, asks to find a subtree t in T such that P is homeomorphic to t. Now, two trees T_1, T_2 are said to be homemorphic if one can remove degree 2 vertices from T_1, T_2 such that T_1 and T_2 are isomorphic. Another recent work [123] considers homeomorphic alignment of "weighted" but unlabeled trees. Here the goal is to obtain a homeomorphic mapping between vertices of two trees such that the differences between the weights of "aligned" edges is minimized. While being related to our approach, the method described in [123] is not applicable to our problem as the trees they consider are not leaf labeled. We refer the reader to [22] for a general and gentle overview of further related work on tree edit distance, tree alignment and tree inclusion.

The main technical contribution of this part of the thesis is a novel deterministic mirrortree algorithm that directly compares tree topologies. The algorithm is optimal within the constraints we impose and is provably efficient. We compare our algorithm with the most recent, state-of-the-art heuristic search approach [66] that aims to maximize the similarity between distance matrices, where distances reflect lengths of shortest paths in neighbor-joining trees. In our comparisons we use precisely the same trees to be able to juxtapose a distance matrix-based heuristic search method to our topology-based, deterministic method without introducing further biases.

5.2 Preliminaries and notations

Let $T = (V, E, w)$ be a tree with weighted edges as given by a non-negative weight function $w : E \to \mathbb{R}_+$. We denote the leaves of T by $L = \{\ell_1, ..., \ell_n\}$, the internal nodes of T (excluding the root) by $U = \{u_1, ..., u_m\}$, and the root of T by r. In particular let n be the number of leaves and m be the number of internal vertices without the root. Note that a tree T is binary and rooted if and only if $\deg(r) = 2$ and $\deg(u) = 3$ for all internal vertices $u \in U$; this will imply that $m = n - 2$ and $|E| = 2n - 2$. In our setting, edge weights $w(v_i, v_j)$ reflect the evolutionary distance between adjacent vertices v_i, v_j. Note that leaves refer to gene products whereas internal vertices can be

interpreted as speciation and/or duplication events. For a given vertex $v \in V$, we define $\theta(v)$ as the evolutionary distance between the root and v. In other words, $\theta(v)$ is the sum of the edge weights in the unique path from the root to v. In rooted trees, there is a natural partial order

$$v_i \leq v_j \quad \Leftrightarrow \quad v_i \text{ is an ancestor of } v_j \tag{5.1}$$

on the vertices of T. Hence, the edges have a natural orientation and each vertex v_i induces a unique subtree $T(v_i)$. This partial order is crucial for our algorithm—which cannot be applied to unrooted trees in a straightforward manner. For processing unrooted (e.g. neighbor-joining) trees, consider the pair of proteins/domains (one from each tree) which are known to interact. We root the two trees at these vertices in order to apply our algorithm. Provided such a pair exists (which is typically the case), our algorithm optimally aligns the trees as it does not assume any order among the many sibling vertices. In a tree T which is rooted at r, we call vertex u the parent of a vertex v if u and v are connected by an edge and u is closer to r than v. The *height* of a rooted tree is defined as $\max\{d(r, \ell_i) \mid i = 1, ..., n\}$ where $d(v_1, v_2)$ is the length of the shortest path between vertices v_1 and v_2 without considering edge weights, that is the maximum (unweighted) distance of the root to a leaf. We denote a bijection (i.e. a one-to-one and onto alignment) of subsets of vertices of T, T' by $\mathcal{M}[T, T']$ and write

$$M := \{(v, w) \in T \times T' \mid \mathcal{M}(v) = w\} \tag{5.2}$$

for the pairs of mapped vertices. Note that in such a bijection, not all vertices of T are necessarily mapped to a vertex in T' and vice versa. We refer to vertices which are not mapped as *deleted* by $\mathcal{M}[T, T']$. We only consider alignments which satisfy the following: (1) the alignment preserves the ancestor relationship of T and T'; (2) only leaves with identical labels are mapped onto one another; (3) upon deletion of vertices, where deletion of an internal vertex v leads to new edges joining the parent of v with the children of v, the two tree topologies are isomorphic. Among the alignments satisfying the above conditions, we compute the alignment that has maximum score.

For a formal definition of our scoring scheme, consider the internal vertices of T and T' that are deleted. Among them, we distinguish between vertices v that have descendants x which are not deleted. We write N_I for such vertices. We write N_T for the remaining deleted vertices. Note that each vertex $v \in N_T$ makes part of a subtree of T which has been deleted as a whole. The score of

the alignment is then defined as

$$S(\mathcal{M}[T,T']) = \sum_{(v,v')\in M} S_M(v,v') + \sum_{v\in N_I} S_{N_I}(v) + \sum_{v\in N_T} S_{N_T}(v). \quad (5.3)$$

The individual score functions S_M, S_{N_I} and S_{N_T} will be formally defined in the Methods section. Note that eventually we set $S_{N_I}(v)$ and $S_{N_T}(v)$ to zero for the purpose of our experiments, but here we include them in the formulation above for completeness. As noted above, our algorithm, which maximizes the overall score of the alignment, can be viewed as an extension of the standard *tree edit distance* algorithm for unweighted trees (e.g. [141]), to those with edge weights. Determining the tree edit distance is NP-complete [163] (in fact MAX-SNP-hard [162]). Since the instances treated here are too large (trees have up to more than 200 leaves) we have to impose reasonable constraints when aiming at fast, polynomial-runtime solutions. Motivated by test runs (see numbers referring to $C_{1,2}$ in the Results and Discussion section), we chose to impose the additional constraint that a vertex u and its parent v cannot be deleted at the same time without deleting the entire subtree rooted at v. That is we disallow to have both a parent v and a child u in N_I. Note, however, that deletion of two internal siblings is permissible—we found that such deletions can lead to favorable alignments. As the operation of deleting entire subtrees does not lead to runtime issues, does not perturb the topology of the remaining trees and also reflects the biologically reasonable assumption that interaction can be lost for entire subtrees, we allow it without additional restrictions.

5.3 Methods

Given two rooted weighted-edge trees T and T', our algorithm *aligns* the trees by mapping a subset of leaves of T to a subset of leaves of T'. In order to obtain this mapping, a series of (1) individual vertex deletions or (2) subtree deletions (with specific penalties) are performed on each tree with the goal of obtaining two isomorphic trees $T_1 = (V_1, E_1, w_1)$ (from T) and $T_1' = (V_1', E_1', w_1')$ (from T_1'); Figure 5.1 shows two such rooted trees that are isomorphic; it also shows a mapping between the leaves. The specifics of vertex and subtree deletions on a tree $T = (V, E, w)$ are as follows.

1. Deleting an internal vertex v also deletes the edge (u, v), where u is the parent of v. Furthermore, it connects each child x of v to u by deleting the edge (v, x) and creating a new edge (u, x). The weight of this new edge, $w(u, x)$ is set to $w(u, v)+w(v, x)$. As mentioned earlier, it is not possible to delete both a node v and its parent u from T.

2. Deleting an entire subtree rooted at an internal vertex v deletes all descendants of v and their associated edges.

In the remainder of this section, we will discuss the costs of the above deletion operations and the scores of the mapped vertices. As mentioned earlier, the overall score of the mapping will be the sum of the scores of the mapped vertices and the scores (negative costs) of the the deletion operations.

Scoring Scheme

Let T_1 and T_1' be the isomorphic trees which result from performing a series of deletion operations on T and T'. The isomorphism $\Phi : T_1 \rightarrow T_1'$ implies an alignment $\mathcal{M}[T, T']$ between the original trees T, T'. Let L_1, L_1' denote the sets of leaves that are mapped in T and T' respectively; because the mapping is a bijection, we must have $|L_1| = |L_1'|$. We write $\mathrm{SP} := \{(l, l') \mid l \in L, l' \in L', (l, l') \in M\} \subset \mathcal{M}[T, T']$ for the set of mapped pairs (we require that mapped leaves have identical labels hence the naming SP for 'species'.

Recall that a mapping of two trees may involve deleting internal vertices or entire subtrees. We now distinguish between two types of internal vertex deletions.

1. [Isolated Deletion:] deletion of only one child v of a vertex u. Let further x_1, x_2 be the two children of v. Isolated deletion of v also implies to also delete edges $(u, v), (v, x_1), (v, x_2)$ and create new edges $(u, x_1), (u, x_2)$. Note that after deletion v has 3 children.

2. [Parallel Deletion:] deletion of both children (say x and y) of a vertex v. This implies deletion and creation of edges in a fashion analogous to that for isolated deletion. Note that after deletion v has 4 children.

Accordingly, we further distinguish between isolated deleted vertices $N_{I,iso}$ and vertices which became deleted in parallel $N_{I,par}$ such that $N_I = N_{I,iso} \dot{\cup} N_{I,par}$. The idea behind distinguishing between isolated and parallel deletion is that parallel deletion reflects greater perturbation of tree topology at the same evolutionary point in time, and is less likely to occur. For a given mapping $\mathcal{M}[T, T']$ let $\mathrm{E}_S(\mathcal{M}) := \{(u, v) \mid v \in N_{I,iso}\}$ be the set of edges which join isolated deleted vertices with their parents. Analogously, $\mathrm{E}_P(\mathcal{M})$ is the set of edges that join deleted siblings with their parent. See figure 5.2 for examples of isolated and parallel deletions.

Given a pair of mapped leaves $\tilde{\ell}_1, \tilde{\ell}_2 \in \mathrm{SP}$ their alignment score, $\kappa(\tilde{\ell}_1, \tilde{\ell}_2)$ is defined as

$$\kappa(\tilde{\ell}_1, \tilde{\ell}_2) = C - |\theta(\tilde{\ell}_1) - \theta(\tilde{\ell}_2)|$$

where C is a positive constant, providing a positive contribution to the overall score because of the mapping of two leaves with the same label while we subtract the difference between the distances of $\tilde{\ell}_1$ and $\tilde{\ell}_2$ from the root for penalizing the mapping between two leaves which have topologic differences.

The total score $\mathcal{S}$ of an alignment $\mathcal{M}[T, T']$ as per the above definition is fully specified by

$$\begin{aligned}\mathcal{S}(\mathcal{M}[T,T']) = &\sum_{(\tilde{\ell}_1,\tilde{\ell}_2)\in \mathrm{SP}} \kappa(\tilde{\ell}_1,\tilde{\ell}_2) \\ &- \sum_{e_s\in E_{iso}(\mathcal{M})} E\cdot w(e_s) - \sum_{e_p\in E_{par}(\mathcal{M})} F\cdot w(e_p)\end{aligned} \tag{5.4}$$

where, with respect to the formulation in (5.3), the term in the first row is for $\sum_{v,v'\in M} S_M(v,v')$, the second row is for $\sum_{v\in N_I} S_{N_I}(v)$ and $\sum_{v\in N_T} S_{N_T}(v)$ is zero. E and F are user-defined constants that respectively penalize *isolated deletion* and *parallel deletion* of edges. Note that this penalty is proportional to the length of the edges joining the deleted vertices with their parents—deletion of longer edges leads to a more severe perturbation of topology hence is more severely penalized. We set the cost of deleting a subtree (i.e. S_{N_T}) to 0. Note, however, deleting subtrees is implicitly penalized by disregarding any potential good mappings of leaves in them.

Given the above score function, the *gene tree alignment problem* can be formally stated as follows.

Gene Tree Alignment Problem

Given two rooted weighted-edge trees T, T', determine subsets of leaves $L_1 \subset L, L'_1 \subset L'$ of equal size such that the corresponding subtrees can be transformed by isolated and parallel deletion and subtree removal operations into trees T_1, T'_1, for which there is an isomorphism $\Phi : T_1 \to T'_1$ that maximizes $\mathcal{S}(\mathcal{M}[T, T'])$.

A Dynamic Programming Solution

The gene tree alignment problem can be efficiently solved by a dynamic programming algorithm. Our algorithm runs in $O((|V|\cdot|V'|)$ time for two binary, rooted trees T, T' with vertex sets V, V'. In general, our strategy can be applied to arbitrary rooted trees with bounded maximum degree, $\Delta_{\max}$. Note that by allowing to delete internal vertices (i.e. contract the edges), the number of children of an internal vertex will be still bounded by a constant (≤ 4).

Initialization As a first step, we remove all leaves that refer to species that are unique to each tree. Let $n = |V|$ and $n' = |V'|$. For every pair of vertices $v_i \in V$ and $v'_j \in V'$ (i.e. for every $i = 1, \cdots, n$ and $j = 1, \cdots, n'$), we compute the maximum alignment score for the subtrees rooted at v_i from T (i.e. $T(v_i)$) and v'_j from T (i.e. $T'(v'_j)$). We denote the maximum alignment score for $T(v_i)$ and $T'(v'_j)$ by S_{ij}. Note that the computation of the maximum alignment score between rooted subtrees induce a mapping between their leaves.

In our dynamic programming algorithm, we handle the "base" cases, where one (or both) of $T(v_i)$ or $T(v'_j)$ have 3 or fewer leaves, as follows.

- If both $v_i \in V$ and $v'_j \in V'$ are leaves, then by definition, $S_{ij} = \kappa(v_i, v'_j)$.
- Without loss of generality, if v_i is a leaf and v'_j is an internal vertex, $S_{ij} = \max(S_{ij_1}, S_{ij_2})$, where j_1 and j_2 correspond to the children of v'_j.
- The remainder of the base cases have both v_i and v_j as internal vertices and are solved through exhaustive evaluation of all possible alignments.

Recursion internal vertices, each with at least 4 descendants, S_{ij} will be computed through recurrence equations. These equations are based on the alignment scores between subtrees rooted at the children (or grandchildren) of v_i and v'_j. Let $i_1(j_1)$ and $i_2(j_2)$ be the children of the vertex $v_i(v'_j)$. Also, let i_{11}, i_{12} be the children of i_1, and i_{21}, i_{22} be the children of i_2. Similarly, let j_{11}, j_{12} be the children of j_1, and j_{21}, j_{22} be the children of j_2. We first give a high level description of the recurrence equation. Suppose that the maximum alignment score between any subtree in $T(v_i)$ and any subtree in $T'(v'_j)$ has already been computed. In order to compute the alignment score S_{ij}, we consider several cases: we can either delete one or both subtrees rooted at the children of v_i and v'_j (deleting an entire subtree) or align the subtrees rooted at the children of v_i and v'_j to each other. We can also delete one of the children of v_i (either i_1 or i_2) together with one of the children of v_j (either j_1 or j_2) and align the three resulting subtrees in $T(v_i)$ to a *permutation* [2] of the ones in $T'(v'_j)$. Finally, we have to consider the case where both children of the root (i.e. i_1 and i_2 in $T(v_i)$, and j_1 and j_2 in $T'(v'_j)$) are deleted. In this case we align four subtrees in $T(v_i)$ (rooted at $i_{11}, i_{12}, i_{21}, i_{22}$) to a permutation of the four resulting subtrees in $T'(v'_j)$. The optimal alignment score of S_{ij} will thus be the *maximum* alignment score provided by all of the cases above.

[2] We have to consider all the permutations because the trees are unordered (i.e. the order of siblings of an internal vertex is unimportant).

Let $e(v)$ denote the penalty for isolated deletion of an internal vertex v, which is the product of the constant E and the weight of the edge between v and its parent (see Scoring Scheme section). Also, let $f(v)$ denote the penalty for parallel deletion of both children of an internal vertex v. $f(v)$ was defined as a constant F times the total weight of the edges that connect v to its children. The recurrence equation for S_{ij} thus becomes the following

$$S_{ij} = \max \begin{cases} 0 \text{ (deleting both subtrees from each tree)} \\ \left\{ \begin{matrix} S_{i_1 j_1} + S_{i_2 j_2} \\ S_{i_1 j_2} + S_{i_2 j_1} \end{matrix} \right\} \text{regular cases} \\ \left\{ \begin{matrix} S_{i j_1} \\ S_{i j_2} \\ S_{i_1 j} \\ S_{i_2 j} \end{matrix} \right\} \text{deleting one subtree from each tree} \\ \max \left\{ \begin{matrix} S_{i_{11} j_2} + S_{i_{12} j_{11}} + S_{i_2 j_{12}} \\ S_{i_{11} j_2} + S_{i_{12} j_{12}} + S_{i_2 j_{11}} \\ S_{i_{12} j_2} + S_{i_{11} j_{11}} + S_{i_2 j_{12}} \\ S_{i_{12} j_2} + S_{i_{11} j_{12}} + S_{i_2 j_{11}} \end{matrix} \right\} - e(i_1) - e(j_1) \\ \max \left\{ \begin{matrix} S_{i_{21} j_2} + S_{i_{22} j_{11}} + S_{i_1 j_{12}} \\ S_{i_{21} j_2} + S^{i_{22} j_{12}} + S_{i_1 j_{11}} \\ Si_{22} j_2 + S^{i_{21} j_{11}} + S_{i_1 j_{12}} \\ S_{i_{22} j_2} + S_{i_{21} j_{12}} + S_{i_1 j_{11}} \end{matrix} \right\} - e(i_2) - e(j_1) \\ \max \left\{ \begin{matrix} S_{i_{21} j_1} + S_{i_{22} j_{21}} + S_{i_1 j_{22}} \\ S_{i_{21} j_1} + S_{i_{22} j_{22}} + S_{i_1 j_{21}} \\ S_{i_{22} j_1} + S_{i_{21} j_{21}} + S_{i_1 j_{22}} \\ S_{i_{22} j_1} + S_{i_{21} j_{22}} + S_{i_1 j_{21}} \end{matrix} \right\} - e(i_2) - e(j_2) \\ \max \left\{ \begin{matrix} S_{i_{11} j_1} + S_{i_{12} j_{21}} + S_{i_2 j_{22}} \\ S_{i_{11} j_1} + S_{i_{12} j_{22}} + S_{i_2 j_{21}} \\ S_{i_{12} j_1} + S_{i_{11} j_{21}} + S_{i_2 j_{22}} \\ S_{i_{12} j_1} + S_{i_{11} j_{22}} + S_{i_2 j_{21}} \end{matrix} \right\} - e(i_1) - e(j_2) \\ S_{i_{11} \pi_1} + S_{i_{12} \pi_2} + S_{i_{21} \pi_3} + S_{i_{22} \pi_4} - f(v_i) - f(v_j) \end{cases} \tag{5.5}$$

where the *permutation* $\pi = \pi_1 \pi_2 \pi_3 \pi_4$ ranges over all permutations of $\{j_{11}, j_{12}, j_{21}, j_{22}\}$. Note that some cases are redundant but are still represented here for the sake of clarity. In case that several

options yield the same, optimal score, the algorithm picks the first observed one.

Now, given r and r', the roots of T and T', respectively, the alignment score $S_{r_1 r_2}$ (i.e. the maximum alignment score of the rooted trees) can be computed using the above recurrence equation, providing a solution to the gene tree alignment problem. It is quite straightforward to prove that our algorithm correctly computes the maximum alignment score through a (strong) induction on the *sum* of the heights of the rooted trees. Note that the scores of internal vertex alignments can be computed through the scores of the alignments between their (grand)children and the recurrence precisely serves to satisfy the constraints. The base of the induction is trivial. If the minimum height of the trees is zero (i.e. one of the trees is just a single leaf), the optimal value of the alignment can be found using the definitions and simple case analysis. Given the subtrees $T(v_i)$ and $T'(v'_j)$, with heights h and h', respectively, we assume the induction hypothesis, that for all pairs of subtrees $T(v_p)$ and $T'(v'_q)$ with heights h_p and h_q such that $h_p + h_q < h + h'$. It is easy to verify by case analysis that all cases in the recurrence equation will be reduced to a case in which the sum of the heights of the aligned (grand) children will be less.

Evaluation Criteria. We determine the maximum number of members of the two protein domain families under consideration that can be paired by following [66]: for each species we determine the paralogous members of the domain in the two trees that can be paired with one another (that is both members reside in the same protein) and determine the maximum number of pairs that can result from the respective potential pairings. Summing up these numbers yields the maximal size of a *correct* mapping. By the usual conventions, we denote this value as P. In other words, P is the size of the correct pairing. Among the P many potential correct pairs, we determine the ones which were inferred by the algorithm in use and refer to them as "true positives", TP. Similarly, the number of domain pairs computed, where the respective members are not from the same protein, is referred to as "false positives", FP. Recall (Sensitivity) is defined as $Rec = TP/P$ and Precision (Positive Prediction Rate) is defined as $Prec = TP/(TP + FP)$. Note that Recall is referred to as Accuracy in [66]. We determine Precision and Recall for each pair of trees individually. Values displayed in Figure 5.3, Table 5.1 and Figure 5.4 (see the Results section) are average values for all 488 co-evolving tree pairs resp. for all tree pairs satisfying the respective criteria.

Method	RP	Recall	RelRec	Precis
C_0	0.546	0.330	0.557	0.447
C_1	0.610	0.378	0.586	0.475
$C_{\{1,2\}}$	0.612	0.377	0.581	0.471
C_{ser}	0.638	0.373	0.556	0.444
C_{Opt}	0.612	0.380	0.588	0.479

Table 5.1: Evaluation of different scoring schemes. C_{Opt} is 'TreeTopology' in the other figures.

5.4 Results

Data Source and Alternative Methods. We benchmarked our algorithm against the most recent heuristic search method [66] for determining a mapping in the presence of paralogs on the large-scale data corpus described in the same study. This data set contains multiple alignments for 604 yeast protein domains among which 488 domain pairs are known to co-occur in the same protein. Those 488 domain family pairs are considered to be a particularly tough test [66] due to the presence of approximately 6 paralogs per species on average. For all interacting domain family pairs, neighbor-joining trees were computed, using ClustalW [142] and the trees were rooted at the domains which are known to interact.

Tree Constraints. In order to appropriately assess the contribution of the different tree constraints as outlined in the Methods section, we evaluated our algorithm by not allowing to delete internal nodes (C_0), allowing isolated node deletion (C_1) as well as further allowing parallel deletion of two sibling nodes ($C_{1,2}$), see Fig. 5.2(c) for an example. We also include the test case C_{ser} where we allow for deletion of a parent and a child, simultaneously. In all the above cases, we allow deletion of subtrees as a whole without penalty. The specific scores for these cases are as follows: $C_0 : C = 1, E = \infty, F = \infty$, $C_1 : C = 1, E = 0, F = \infty$ and $C_{1,2} : C = 1, E = F = 0$.

Among the cases above $C_{1,2}$ gives the best results (see Table 5.1 and further comments below), implying that paralell deletions are beneficial. We experimented with several values of E and F, and noticed that it may be beneficial to impose a large penalty for parallel deletions in contrast to a relatively small penalty for isolated deletions. We concluded that an optimal choice of parameters would be $E = 2, F = 50$ (referred to as C_{Opt}), when $C = 1$. Note that the exact value of C is the function of neighbor-joining trees resulting from ClustalW multiple alignments alone—for different settings absolute values need to be put into perspective with orders of magnitude of edge weights of

the trees under consideration.

As outlined in the Methods section, inducing tree constraints considerably reduces the search space, thereby allowing for an efficient and deterministic method. Given a pair of trees, let CP ("Constraint Positives") denote the maximum number of *correctly* paired domains over all possible *alignments* of the trees. Note that one can compute, CP for any given pair of trees, by running our algorithm with a scoring scheme which assigns 1 to correctly paired domains and not penalizing any operation which the constraints allow us to do. [3] We compute $RP = \frac{CP}{P}$ ("Relative Positives") as the fraction of pairings that can still be inferred, and which measures the reduction of search space size due to imposing constraints. We further compute $\mathrm{RelRec} = \frac{TP}{CP}$ ("Relative Recall") as a recall value which reflects how many of the correct pairings possible were inferred by the algorithm in question. The good RelRec values the tree topology approach achieves (0.59 vs. 0.55 for the matrix-based approach, note that for the matrix-based approach this coincides with Recall since it does not impose any constraints), indicate that losses in Recall are due to imposing constraints, but not necessarily due to the scoring scheme.

Figure 5.3 presents numbers of all 488 tree pairs for a canonical baseline procedure, which randomly pairs as many domain family members per species as possible, the heuristic matrix-based approach (MatrixHeuristic) and the deterministic tree-topology based approach (TreeTopology $= C_{opt}$). Table 5.1 furthermore presents numbers resulting from usage of different tree constraints. Following [66], we also separate tree pairs according to the numbers of leaves of the larger tree and the product of the numbers of leaves of the two paired trees which, according to [66], quantifies search space size. See Figure 5.4 for the respective results.

5.5 Discussion

Runtime. The possibly most striking advantage of our topology-based approach is the drastic reduction of runtime—we can compute mappings for the 488 interacting domain families in roughly 1 minute on a single CPU - in comparison to 730 hours on MareNostrum [4] needed for the Metropolis search performed by [66]. Note that there are rapidly growing large-scale phylogenetic databases such as ENSEMBL (`http://ensembl.org`) or PhylomeDB (`http://phylomedb.org`),

[3] This scoring scheme assumes knowledge we are not allowed to use in the algorithm; we use this knowledge for the purpose of evaluation here.

[4] MareNostrum is a supercomputer of the Barcelona Supercomputing Center, one of the largest machines in the world dedicated to science [66, p. 10].

whose growth is further accelerated by next-generation sequencing projects (as of 12th August, 2011, PhylomeDB contains $482,274$ phylogenetic trees). The reduction in runtime delivered by our approach certainly overcomes a major obstacle—we render large-scale mapping and, as a consequence, comparison of paralog-rich gene trees feasible. Note that this reduction has become possible by imposing both computationally and biologically reasonable constraints on the search space while at the same time allowing for an efficient scheme to find the global optimum within these constraints.

Search Space Size / Recall. Comparing C_{Opt} with the method of [66] (Heuristic) overall, clearly, [66] achieve best recall. As pointed out above, this comes as no surprise since we cannot explore pairings that contradict the topologies of the paired trees. Quite surprisingly though, although usage of tree topology and neighbor-joining trees in particular have been discussed rather controversially [151], we find that still the majority of pairings (54.6% with the strictest constraints and 61.2% for allowing isolated and parallel deletion) can be determined by a topology-based approach. These numbers may put usage of neighbor-joining tree topology in mirrortree approaches into a general perspective. Moreover, note that the fraction of correct domain pairs computed by our method over that of the heuristic search method is about 0.7 ($= \frac{TP(C_{opt})}{TP(Heuristic)} = \frac{\mathrm{Recall}(C_{opt})}{\mathrm{Recall}(Heuristic)} = \frac{0.38}{0.55}$) which is more than what was to be expected by reduction of the search space ($\frac{CP(C_{opt})}{CP(\mathrm{Heuristic})} = \frac{CP(C_{opt})}{P(\mathrm{Heuristic})} = RP(C_{opt}) = 0.61$) which points out that we compensate search space reduction by a more effective search strategy. This becomes reflected by the better RelRec values of C_{opt}.

Precision Precision favors the topology-based approach, at least on larger (combinations of) trees (see column Prec in all three tables). Better precision reflects a larger fraction of the correct domain pairs among the pairs inferred overall. We achieve slightly better values in terms of Precision than [66], see Prec in Figure 5.3. See also Figure 5.4 for a comparison with respect to search space size-related differences. While [66] achieve excellent values on pairs of smaller trees, we outperform their approach on larger trees, with the most obvious differences on pairs of trees where the product of the numbers of leaves is large.

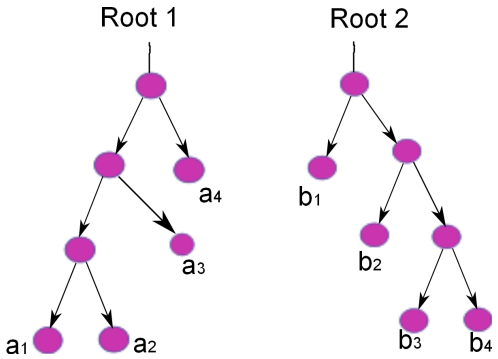
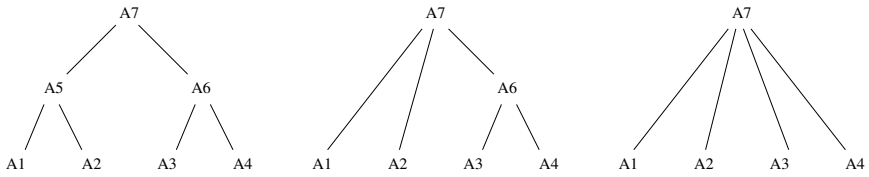


Figure 5.1: Two isomorphic trees are shown as an example in this figure. The leaves of the left tree are labeled with a_1, a_2, a_3, a_4 while the leaves of the tree on the right are labeled with b_1, b_2, b_3, b_4. A possible mapping between the leaves that respect the tree topology is $(a_1, b_3), (a_2, b_4), (a_3, b_2), (a_4, b_1)$.



(a) No deletion (b) An isolated deletion: A_5 (c) Parallel deletions of two nodes: A_5 and A_6

Figure 5.2: A gene tree (a), with an isolated node deletion, A_5 (b) and a parallel deletion of the nodes A_5 and A_6) (c).

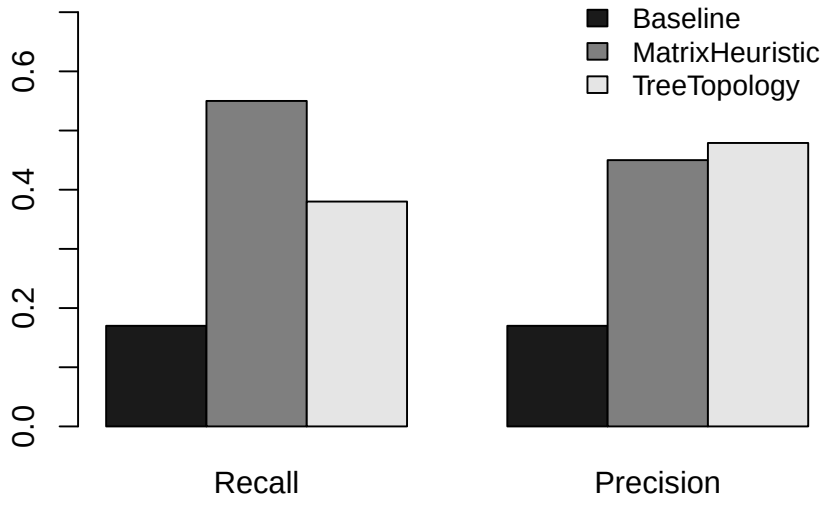


Figure 5.3: Recall and Precision for the heuristic matrix-based approach (MatrixHeuristic, [66]) and the deterministic, tree-topology based approach (TreeTopology $= C_{opt}$). Baseline is determined as randomly pairing as many protein domain family members as possible. Runtimes for MatrixHeuristic and TreeTopology are 730 hours and 1 minute respectively.

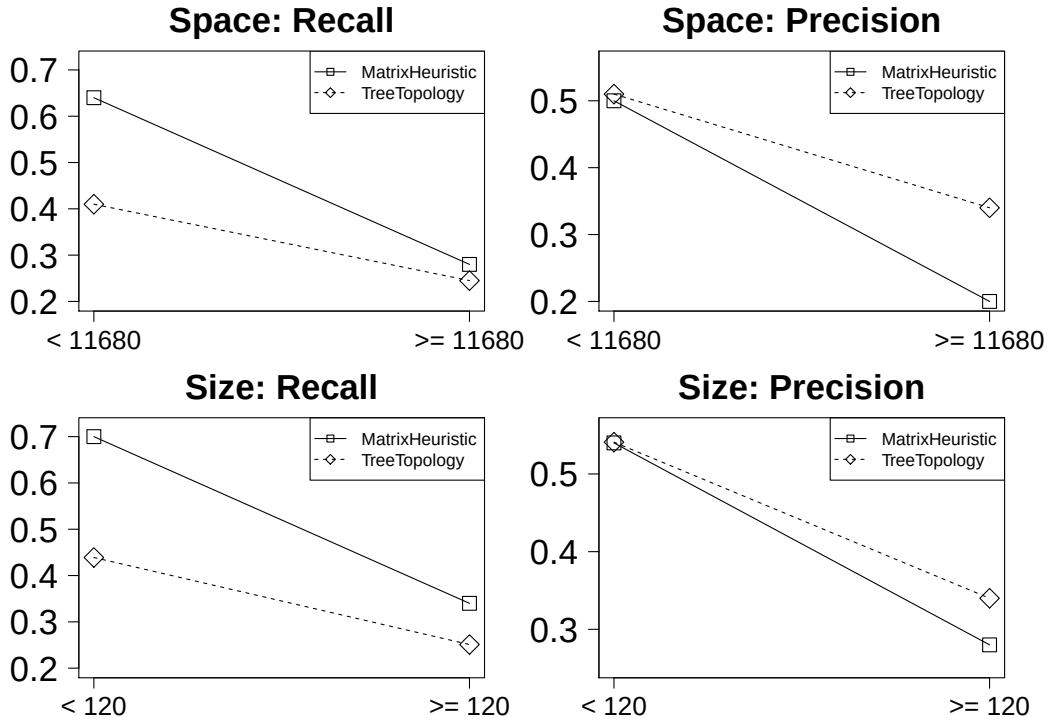


Figure 5.4: The comparison of our method with the heuristic search method reveals favorable results for large trees (bottom row), x-axis indicates the size (number of leaves) of the larger tree of the two trees paired and in particular for large search spaces, that is for Space $\geq$11680 where Space is the product of the number of leaves of the two trees paired.

Chapter 6

Motif counting in protein-protein interaction networks

In the previous chapter, we discussed our computational method for identifying pairs of interacting proteins or domains. We shift our focus in this chapter to the study of networks of protein interactions themselves and their topological features. Recent research has revealed that many protein-protein interaction (PPI) networks share global topological features. Similarities between PPI networks of several organisms have been observed with respect to their degree distribution, k-hop reachability, betweenness and closeness [18, 24, 53, 118]. Because direct measures for comparing two networks, such as the minimum number of edges and vertices to be deleted to make two networks isomorphic are NP-hard to compute, such topological features have been used to "measure" how similar any given pair of networks could be. Two networks which have similar global features can have significant differences in terms of local structures they include: e.g., one of them may include a specific subgraph many more times than the other. Thus it is important to be able to count the "number of occurrences" of specific subgraphs in networks as means of detecting whether two networks are similar or not.

A subgraph that occurs much more frequently in a biomolecular network G than one in a "random" network or a "typical" network R whose global properties are similar to those of G is called a network motif of G [108]. Similarly, a subgraph that occurs much less frequently in G in comparison to R is called an anti-motif of G. The motifs were suggested to be recurring circuit elements that carry out key information processing tasks [108] and thus are of considerable interest. Thus, the number of occurences of a specific subgraph can be used as a mean to detect whether it is a motif.

The use of subgraph distribution with up to k vertices to compare PPI networks with artificial networks has been the source of a recent debate. It was argued that the distribution of subgraphs of up to $k = 5$ vertices in the Yeast PPI network is quite different from that of the preferential attachment model [118]. Based on this observation, it was argued that the Yeast PPI network is not a "scale-free" network and the presumed similarity of the Yeast PPI network and the "scale-free" networks in terms of degree distribution is a consequence of sampling errors [50]. Finally, in [53] it was demonstrated that the subgraph distribution of the preferential attachment model and that of the duplication model for $k \leq 6$ can be substantially different and the seed network of the duplication method could be chosen in a way that its subgraph distribution can be made "very similar" to that of the available PPI networks including that of the Yeast.

Although it is possible to make the general distribution of subgraphs in a artificial model (more specifically the duplication model) very similar to that of a specific PPI network, there are a number of subgraphs, for example in the Yeast PPI network, which occur much more frequently than that in the associated random model. These motifs were suggested to be recurring circuit elements that carry out key information processing tasks [108] and thus are of considerable interest.As a result novel computational tools have been developed for counting subgraphs in a network [118, 53] and discovering network motifs [44].

Counting the number of all possible "induced" subgraphs in a PPI network has been proved to be a challenging task. [118] describes how to count all induced subgraphs with up to $k = 5$ vertices in a PPI network. Faster techniques that count induced subgraphs of size up to $k = 6$ [53] and $k = 7$ [44] were developed very recently. The running time of these techniques all increase exponentially with k thus novel algorithmic tools are now needed for counting subgraphs of size $k \geq 8$.

[73] provides an efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. The sampling algorithm is based on a random walk approach which in k (the number of vertices of the subgraph) iterations picks k vertices of the original graph and includes all the edges between the picked vertices. Surprisingly, the experiments show that a few samples is sufficient to achieve an accurate result.

Note that an induced subgraph (more accurately a vertex induced subgraph) of a network G is a subset of the vertices of G together with any edges whose endpoints are both in this subset; i.e. G' is an induced subgraph of G if and only if for each pair of vertices v' and w' in G' and their corresponding vertices v and w in G, either there are edges between both v', w' pair and v, w pair or there are no edges between any of the pairs. For example let G be a fully connected graph of size n. Then a cycle that goes through every vertex in G is not an induced subgraph of G; it is called a

"non-induced" subgraph of G.

All the above techniques consider only induced subgraphs of a given network; there are many more non-induced subgraphs isomorphic to a given topology and thus it is more difficult to count non-induced subgraphs of a network. As a result, there are only a limited number of earlier studies on biomolecular networks that consider non-induced subgraphs [35, 132]. The motivation for considering non-induced subgraphs are clear: available PPI networks are far from complete and error free; the interactions between proteins reported by these networks include both false positives and false negatives. Thus an occurrence of a specific network motif in one network may include additional edges in its occurrence in another network and vice versa.

The specific problem addressed by earlier studies on non-induced subgraphs [35, 132] is not the subgraph counting problem. Rather these papers focus on the "subgraph detection" problem, which aims to respond to queries of the form, does an input network G have a non-induced subgraph G' - where G' is a user specified query subgraph. Subgraph detection problem is somewhat easier than the subgraph counting problem. [35], for example, show how to solve the subgraph detection problem for subgraphs of size $k = O(\log n)$ - much larger than what can be tackled by [118, 53, 44] for subgraph counting - provided that the query subgraph G' is either a simple path, a tree, or a bounded treewidth subgraph. The main tool employed here that makes subgraph detection problem tractable for such subgraphs is the "color coding" technique [11].

Color coding is a combinatorial approach that was introduced to detect simple paths, trees and bounded treewidth subgraphs in unlabeled graphs [11]. It was later applied to subgraph detection in biomolecular networks by [133, 35]. Color coding is based on assigning random colors to the vertices of an input graph. For subgraph detection purposes, it considers only those subgraphs where each vertex has a unique color as a potential answer to a query subgraph. Such "colorful" subgraphs which are isomorphic to the query subgraph can then be detected through efficient use of dynamic programming, in time polynomial with n, the size of the input graph. If the above procedure is repeated sufficiently many times (polynomial with n, provided that the subgraph we are looking for is of size $k = O(\log n)$), it is guaranteed that a specific occurrence of the query subgraph will be detected with high probability.

[13] use the color coding approach to count the number of subgraphs in a given graph G which are isomorphic to a *bounded treewidth graph* H. They give a randomized approximate counting algorithm with running time $k^{O(k)} \cdot n^{b+O(1)}$ where n and k are the number of vertices in G and H, respectively, and b is the treewidth of H. The framework which they use is based on [72] on approximate counting via sampling. Provided that $k = O(\log n)$, the running time of this algorithm

is *super-polynomial* with n, and thus is not practical.

[9] combines the color coding technique with a construction of what is called *Balanced Families of Perfect Hash Functions* to obtain a *deterministic* algorithm to count the number of *simple paths or cycles* of size k in an input graph G with running time $2^{O(k \log \log k)} n^{O(1)}$, still *super-polynomial* in n when $k = O(\log n)$.

Given a network with n vertices, we show how to apply the color coding technique to *count* non-induced trees and bounded treewidth subgraphs with k vertices. We present a randomized approximation algorithm with running time $2^{O(k)} \cdot n^{O(1)}$, which is polynomial in n for $k = O(\log n)$ and thus is faster than available alternatives [13, 9]. Our algorithm is quite efficient in practice; we were able to go beyond what the algorithms presented in [118, 53, 44] achieve, and count, for the first time, *all* possible tree topologies of $8, 9$ and 10 vertices in PPI networks of various organisms such as *S.Cerevisiae* (Yeast), *E.coli*, *H.pylori* and *C.elegans* (Worm) PPI networks available via the DIP database [156], which was released at the time we developed our algorithm.

The distribution of trees of up to 10 vertices provides us powerful means to compare biomolecular networks. We define a novel "normalized treelet distribution", the distribution of the number of occurrences of non-induced trees in a PPI network, normalized by the total number of such treelets, is that it is quite robust. On the well known Yeast PPI network [156], even after random sparsification with bait coverage of 70% and edge coverage of 70% (as suggested by [50]), the normalized tree distribution does not change much. However, differences become noticeable after sparsifying the Yeast PPI network with 50% bait and 50% edge coverage. It is interesting to note that the normalized treelet distributions of the three unicellular organisms we compared, Yeast, *E.coli* and *H.pylori* were all fairly similar; however the distribution of the more complex *C.elegans* was quite different.

In the following section, we present our efficient algorithm that is able to count all trees of size ≤ 10 in PPI networks of various organisms such as *S.Cerevisiae* (Yeast), *E.coli*, *H.pylori* and *C.elegans* (Worm).

We show that the occurences of trees of up to 10 vertices provides us powerful means to compare biomolecular networks. We define a novel "normalized treelet distribution", the distribution of the number of occurrences of non-induced trees in a PPI network, normalized by the total number of such treelets, is that it is quite robust. On the well known Yeast PPI network [156], even after random sparsification with bait coverage of 70% and edge coverage of 70% (as suggested by [50]), the normalized tree distribution does not change much. However, differences become noticeable after sparsifying the Yeast PPI network with 50% bait and 50% edge coverage. It is interesting to note that the normalized treelet distributions of the three unicellular organisms we compared, Yeast,

E.coli and *H.pylori* were all fairly similar; however the distribution of the more complex *C.elegans* was quite different.

6.1 The Subgraph Counting Algorithm

We apply the color coding technique to approximately count the number of non-induced occurrences of each possible tree topology T with $O(\log n)$ vertices in a network G with n vertices. As per [13], this method can be generalized (without much difficulty) to count all non-induced occurrences of each bounded treewidth graph G' in G as well, provided that the treewidth is constant.

Given a graph G with n vertices and a tree T with k vertices, we consider the problem of counting the number of non-induced subtrees of G which are isomorphic to T. Note that we use the standard definition of a tree, i.e. for us, a tree is an unlabeled, connected graph with no cycles. It is unrooted and its vertices are unordered.[1] A tree T is said to be isomorphic to a subtree T' in a graph G if there is a bijection between the vertices of T and the vertices of T' such that for every edge between two vertices a and b of T there is an edge between the vertices a' and b' in T' that correspond to a and b respectively. Such a tree T' is considered to be a non-induced occurrence of T in G.

Note that we allow overlaps between the trees we count, i.e. two occurrences of T, namely T' and T'' may share vertices; in fact the vertex sets of T' and T'' may be identical. We consider T' and T'' distinct occurrences of T provided that the edge sets of T' and T'' are not identical.

Our algorithm counts the number of non-induced occurrences of a tree T with $k = O(\log n)$ vertices in a graph G with n vertices as follows.

1. **Color coding.** Color each vertex of input graph G independently and uniformly at random with one of the k colors.

2. **Counting.** Apply a dynamic programming routine (explained later) to count the number of non-induced occurrences of T in which each vertex has a unique color.

3. Repeat the above two steps $O(e^k)$ times and add up the number of occurrences of T to get an estimate on the number of its occurrences in G.

In what follows, we give the details of the above steps and explain how and why they work.

[1] Thus, for example, consider a tree T with a root vertex a with two children b and c, with b having a single child d. For our purposes T is isomorphic to another tree where b is the root with two children d and a, and a with a single child c. In fact both of these trees are isomorphic to a simple path involving four vertices.

6.2 Color coding step

We note that the color coding step not only works for trees but also bounded treewidth graphs with constant treewidth. Let r be the total number of copies of T in G. We assign a color to each vertex of G from the color set $[k] = \{1, \cdots, k\}$. The colors are assigned to each vertex independently and uniformly at random. It is easy to see that for a particular non-induced occurrence of T in G the probability that all its vertices are assigned unique colors is $p = k!/k^k$, thus the expected number of colorful copies in G is rp.

Let $\mathcal{F}$ denote the family of all copies of T in G. For each such copy $F \in \mathcal{F}$, let x_F denote the indicator random variable whose value is 1 if and only if the copy is colorful in our random k-coloring of $V(G)$, the vertices of G. Let $X = \sum_{F \in \mathcal{F}} x_F$ be the random variable counting the total number of colorful copies of T. By linearity of expectation, the expected value of X is $E(X) = rp$.

It is possible to estimate the variance of X as follows. Note,first, that for every two distinct copies $F, F' \in \mathcal{F}$, the probability that both F and F' are colorful is at most p (and in fact strictly smaller unless both copies have exactly the same set of vertices), implying that the covariance $Cov(x_F, x_{F'})$ satisfies

$$Cov(x_F, x_{F'}) = E(x_F x_{F'}) - E(x_F)E(x_{F'}) \leq p.$$

Therefore, the variance of X satisfies

$$\begin{aligned} Var(X) &= \sum_{F \in \mathcal{F}} Var(x_F) + \sum_{F \neq F' \in \mathcal{F}} Cov(x_F, x_{F'}) \\ &\leq rp + r(r-1)p = r^2 p. \end{aligned}$$

It follows that if Y is the average of s independent copies of X (obtained by s independent random colorings), then

$$E(Y) = E(X) = rp$$

and

$$Var(Y) = Var(X)/s \leq r^2 p/s.$$

Therefore, by Chebyshev's Inequality, the probability that Y is smaller than (or bigger than) its expectation by at least ϵrp is at most

$$\frac{r^2 p}{\epsilon^2 r^2 p^2 s} = \frac{1}{\epsilon^2 ps}.$$

In particular, if $s = \frac{4}{\epsilon^2 p}$ this probability is at most $1/4$.

In case we wish to decrease the error probability, we can compute Y t times independently and let Z be the median. The probability that the median is less than $(1-\epsilon)rp$ is the probability that at least half of the copies of Y computed will be less than this quantity, which is at most

$$\binom{t}{t/2}4^{-t} \leq 2^{-t}.$$

A similar estimate holds for the probability that Z is bigger than $(1+\epsilon)rp$. Therefore, if $t = \log(1/\delta)$ then with probability $1-2\delta$ the value of Z will lie in $[(1-\epsilon)rp, (1+\epsilon)rp]$. Note that the total number of colorings in the process is

$$O(\frac{\log(1/\delta)}{\epsilon^2 p}) = O(\frac{e^k \log(1/\delta)}{\epsilon^2}).$$

Our estimate for r is, of course, $Z/p = Zk^k/k!$.

6.3 Counting step

Given a random coloring of the input graph G with k colors, we present a dynamic programming algorithm to compute the number of colorful subgraphs of G which are isomorphic to the query tree T.

To give a flavor of our algorithm, we first present for the case in which the query graph is a single path of length k. For each vertex v and each subset S of the color set $\{1, \cdots, k\}$, we aim to record the number of colorful paths for which one of their endpoints is v. Let $C(v, S)$ be the number of such paths, and $col(v)$ be the color of vertex v. Given a color ℓ, for all $v \in V(G)$:

$$C(v, \{\ell\}) = \begin{cases} 1 & \text{if } col(v) = \ell \\ 0 & \text{otherwise.} \end{cases}$$

For each vertex v and color set S where $|S| > 1$, we have

$$C(v, S) = \sum_{u;(u,v)\in E(G)} C\big(u, S - col(v)\big).$$

Note that the number of single colorful paths of length k would be

$$\frac{1}{2}\sum_{v} C\big(v, \{1, \cdots, k\}\big).$$

As mentioned earlier, we will only describe the counting step for the case T is a tree, however the algorithm we present can be generalized to bounded treewidth graphs with constant treewidth without much difficulty.

As a first step we pick an arbitrary vertex ρ of T and set it as the *root*. We will denote this rooted tree by $\tau(\rho)$. Then we count the number of colorful occurrences of $\tau(\rho)$ in the given graph G as follows.

For each vertex v of the graph G, we compute $c(v, \tau(\rho), [k])$, the number of $[k]$-colorful rooted subtrees with root v, which are isomorphic to $\tau(\rho)$.

The actual number of $[k]$-colorful occurrences of T in G is

$$\frac{1}{q} \sum_{v} c(v, \tau(\rho), [k])$$

where q is equal to the number of vertices u in T, for which the rooted tree $\tau(u)$ is isomorphic to $\tau(\rho)$.

In order to compute $c(v, \tau(\rho), [k])$ for every vertex v in the graph G, we use the following dynamic programming routine.

Let $\tau'(\rho')$ be a subtree of the tree $\tau(\rho)$ with root ρ', we denote the size of $\tau'(\rho')$ by $\nu(\tau'(\rho'))$. For any vertex x in G, and a subset S of the color set $[k]$ with $|S| = \nu(\tau'(\rho'))$, let $c(x, \tau'(\rho'), S)$ be the number of S-colorful subgraphs with root x and color set S, which are isomorphic to $\tau'(\rho')$. We compute $c(x, \tau'(\rho'), S)$ inductively as follows.

The base case where $\nu(\tau'(\rho')) = 1$ is obvious: For any single color set $S = \{a\}$, $c(x, \tau'(\rho'), S)$ is equal to 1 if x has color a, and otherwise is equal to 0.

For the case where $\nu(\tau'(\rho')) \geq 2$, let ρ'' be a vertex connected to ρ' in $\tau'(\rho')$. Removing the edge (ρ', ρ'') partitions $\tau'(\rho')$ into two smaller subtrees, say $\tau'_1(\rho')$ with root ρ', and $\tau'_2(\rho'')$ with root ρ''.

Now for every vertex u connected to x in G, and all set of colors S_1 and $S_2 \subset [k]$ with $|S_1| = \nu(\tau'_1(\rho'))$, $|S_2| = \nu(\tau'_2(\rho''))$, and $S_1 \cap S_2 = \emptyset$ we recursively find $c(x, \tau_1(\rho'), S_1)$ and $c(u, \tau_2(\rho''), S_2)$. The next step is to compute $c(x, \tau'(\rho'), S)$, by using the values of $c(x, \tau_1(\rho'), S_1)$ and $c(u, \tau_2(\rho''), S_2)$ for every u connected to x, and all feasible set of colors S_1 and S_2. This is easily achieved by the fact that

$$c(x, \tau'(\rho'), S) = \frac{1}{d} \sum_{\forall S_1, S_2 : |S_1 \cap S_2| = \emptyset} c(x, \tau_1(\rho'), S_1) \cdot c(u, \tau_2(\rho''), S_2).$$

Here, d is the *over counting factor* and is equal to one plus the number of siblings of ρ'', i.e. vertices connected to ρ', in $\tau'(\rho')$.

# of vertices (k)	# of unlabeled trees	Running time (mins)
7	11	2
8	23	14
9	47	100
10	106	700

Table 6.1: Number of unlabeled tree topologies, and the running time of our algorithm to count them in the Yeast PPI network.

Network	# Vertices	# Edges	Average degree
S.cerevisiae	2345	5609	4.78
E.coli	1441	5871	8.14
H.pylori	687	1351	3.93
C.elegans	2387	3825	3.20

Table 6.2: Number of vertices, edges, and average degree in the PPI networks studied.

Note that the total running time of our algorithm would be polynomial in n. We need to repeat the experiment $O(\frac{e^k \log(1/\delta)}{\epsilon^2})$ times, and each counting step takes $O(2^k \cdot |E|)$ where $|E|$ is the number of edges in the input network. Thus the asymptotic running time of our algorithm is

$$O(|E| \cdot 2^k \cdot e^k \log(1/\delta) \cdot \frac{1}{\epsilon^2})$$

.

6.4 Experimental Results

We tested our algorithm to count non-induced occurrences of subgraphs with $k = 8, 9, 10$ vertices. The table 6.1 shows the number of unlabeled tree topologies for different values of k together with the total running time of our algorithm for counting the non-induced occurrences of these trees in the largest connected component of Yeast PPI network. Note that our algorithm is fast; for $k = 10$, it takes 12 hours to count all tree topologies on a Sun Fire X4600 Server with 64GB RAM, when executed in parallel on 8 dual AMD Opteron CPUs 2.6 Ghz.[2]

[2]We do not provide a direct comparison of this method with others w.r.t. running time as we focus on counting non-induced occurences of motifs whereas all alternatives focus on induced occurences.

The list of different tree topologies of varying k can be obtained from the Combinatorial Object Server Generation website (http://theory.cs.uvic.ca/cos.html).

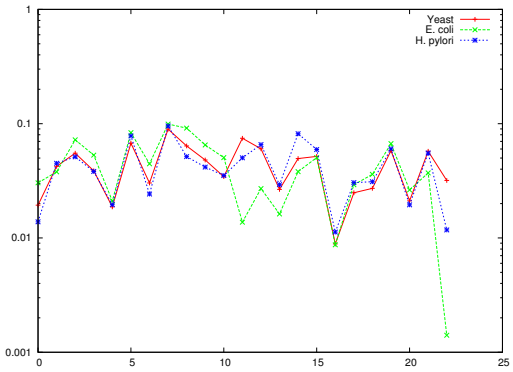
We tested our algorithm on the protein-protein interaction networks of four species; *S.cerevisiae* (Yeast), *E.coli*, *H.pylori*, and *C.elegans* (Worm). Since the PPI networks of these species are far from complete, we focus on the largest connected component of each network. For each PPI network and for all trees of $k = 8, 9, 10$ vertices, we counted the number of non-induced occurrences of each tree topology. The distribution of the number of such subtree topologies (which will be called "treelets") for varying values of k provide means of comparing PPI networks.

Note that the number of vertices, their average degree, etc. vary significantly from one PPI network to the other. Table 6.2 shows the number of vertices and edges of the PPI networks we used in our study. Thus it should be expected that the number of non-induced occurrences of treelets should differ considerably among the networks.

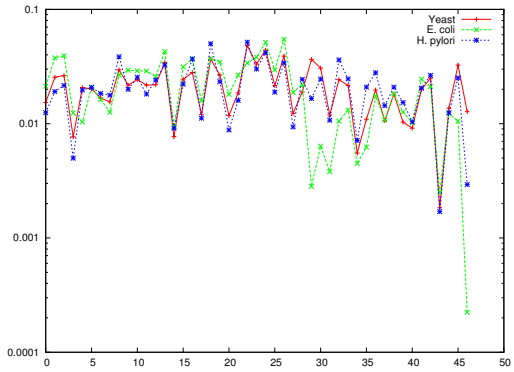
As a result, we *normalize* the treelet distributions of each individual network, for each value of k as follows. For each treelet T of k vertices, consider the *fraction* of the number of occurrences of T in a network G among *total number of occurrences of all possible treelets* of size k in G. The normalized treelet distribution refers to this fractional count of treelets in a given PPI network. We note that as specific fractions of treelets vary by several orders of magnitude, our normalized treelet distributions are all given in logarithmic scale.

6.5 Comparing PPI networks w.r.t. normalized treelet distribution

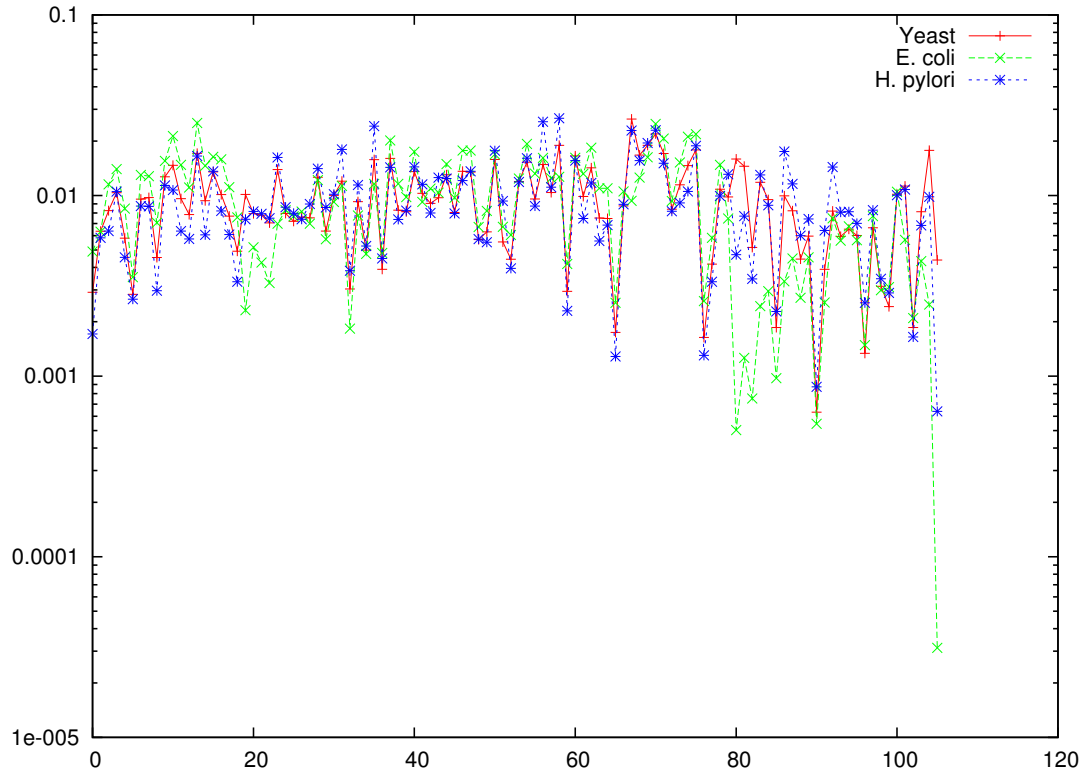
We first discuss how the normalized treelet distributions vary among the most complete PPI networks available via the *Database of Interacting Proteins* [156]. Figure 6.1 compares the normalized treelet distributions of the Yeast, *H.pylori*, and *E.coli* PPI networks (in fact their largest connected components). Although all three PPI networks of unicellular organisms are quite similar with respect to normalized treelet distributions, it is interesting to note that the PPI network of the Yeast appears to be more similar to that of the *H.pylori* in comparison to the *E.coli* PPI network. We also compare the normalized treelet distributions of these three unicellular organisms' PPI network with that of the most complete PPI network of a multicellular organism, *C.elegans* in Figure 6.2. As can be seen, the normalized treelet distribution of *C.elegans* is very different from that of the Yeast, *E.coli* or *H.pylori*.



(a) Treelet distribution of size 8

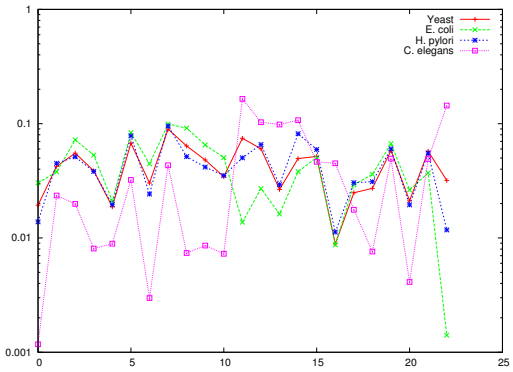


(b) Treelet distribution of size 9

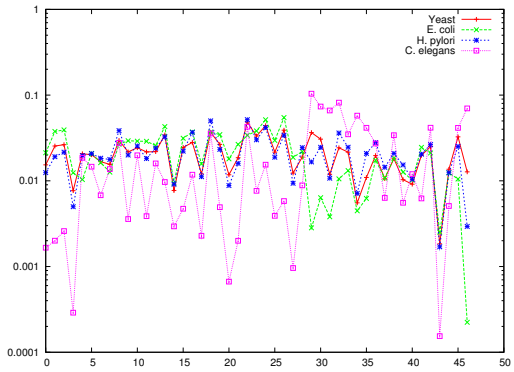


(c) Treelet distribution of size 10

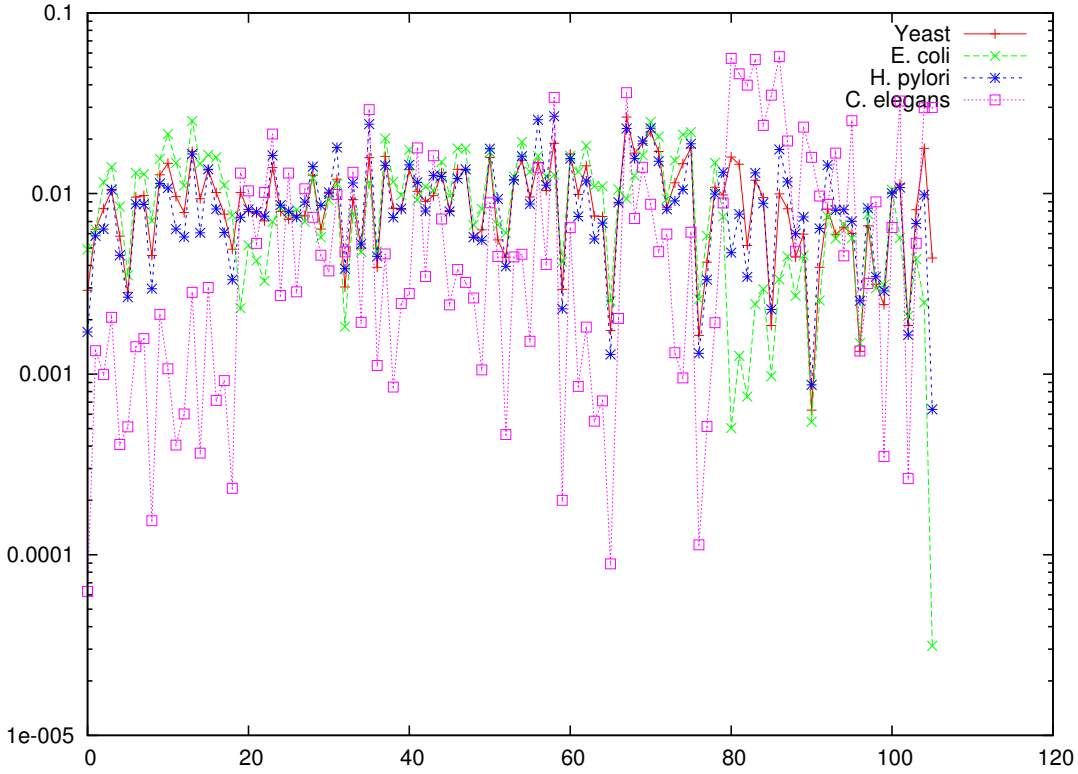
Figure 6.1: Normalized treelet distribution of the Yeast PPI network (Red), *E.coli* (Green) and *H.pylori* (Blue)



(a) Treelet distribution of size 8



(b) Treelet distribution of size 9



(c) Treelet distribution of size 10

Figure 6.2: Normalized treelet distribution of the Yeast PPI network (Red), *E.coli* (Green) , *H.pylori* (Blue) and *C.elegans* (Pink)

Chapter 7

Conclusion

In this thesis, we presented our effort in developing new methods for structural variation discovery in sequenced genomes. We designed a computational pipeline, NovelSeq, to assemble the novel sequence insertions and build maps of insertion by anchoring the sequences back into the reference genome assembly. An important aspect of the NovelSeq framework is that it can be applied as a post-processing step after the completion of read mapping to analyze other types of genetic variation such as SNP and structural variation discovery. Furthermore, we described additional algorithms to discover transposon insertions that are of known importance to genome evolution and genomic variation. These enhancements provide a much needed step towards a highly reliable and comprehensive structural variation discovery algorithm, which, in turn will enable genomics researchers to better understand the variations in the genomes of newly sequenced human individuals, as well as the genome structures of non-human species. We also demonstrated that analyzing a collection of high-throughput sequenced genomes jointly and simultaneously improves structural variation discovery, especially among highly related genomes. We focused on discovering deletions and *Alu* repeats among high throughput paired-end sequenced genomes of family members and showed that the algorithms we have developed for simultaneous genome analysis provide much lower false positive rates in comparison to existing algorithms that analyze each genome independently. Our algorithms, which are collectively call CommonLAW (Common Loci structural Alteration Widgets) aim to solve the maximum parsimony structural variation discovery for multiple genomes problem optimally through a generalization of the maximum parsimony formulation and the associated algorithms introduced in [55, 57, 48] for a single donor genome. We believe that the CommonLAW framework will help studies on multiple, highly related, high coverage NGS genome sequences from members of a family or an ethnic group, tissue samples from one individual (e.g. primary

tumor vs metastatic tumor), individuals sharing a phenotype, etc., by identifying common and rare structural alterations more accurately. The YRI family on which we demonstrate the effectiveness of the CommonLAW framework in the context of de novo *Alu* repeat discovery, or the CEU family whose genomes we analyzed for deletion discovery provide convincing evidence that CommonLAW framework may make a significant difference in large scale projects involving high coverage NGS data. In addition, we believe that our methods can also be adopted to analyzing low coverage NGS data for improving the accuracy provided by available software tools. Note that CommonLAW is already being used as a discovery methods for medical genomic projects such as the Autism project. However there are still open problems. In the second part of the thesis, we have, for the first time, devised a deterministic and efficient, polynomial-runtime mirrortree approach which directly compares the gene trees, and not the distance matrices behind or giving rise to them. We have juxtaposed our approach with the most recent, state-of-the-art matrix-based heuristic search procedure without introducing further experimental biases. Most importantly, our tree topology-based algorithm lists efficiency as its decisive benefit. While recall is better for the heuristic search obviously due to that it does not impose any constraints on the search space, we only incur relatively mild losses. We achieve better results in precision, in particular when both of the mirrored trees are large. This leads us to the overall conclusion that the heuristic method remains the better choice for smaller trees and when runtime is not an issue. In case of larger trees and in particular for large-scale studies, our approach has considerable benefits. Note finally that we have been comparing neighbor-joining trees which have been repeatedly exposed as suboptimal choices of phylogenetic trees. We believe that our approach can gain from improvements in tree quality significantly more than the matrix-based approaches. Note finally that mapping domains can lead to ambiguous results due to that several homologous copies can co-occur in one protein. To resolve such issues is interesting future work. Finally, we presented our algorithm for counting non-induced occurrences of sizable subgraphs in a protein-protein interaction network, provided that the subgraphs in question are in the form of trees or bounded treewidth graphs. We showed how to apply color coding technique to count the number of non-induced occurrences of such subgraphs in time polynomial with n if $k = O(\log n)$, where n is the number of vertices in the network and k is the query size. We used our algorithm to obtain all treelet distributions for $k \leq 10$ of the PPI networks of unicellular organisms (S.cerevisiae, E.coli and H.pylori), which are all quite similar, and a multicellular organism (C.elegans) which is significantly different.

Future directions As we discussed throughout the thesis the problem of structural variation (SV) discovery is considered one of the most important problems in the field. Thus some of the future directions would include improving the existing SV discovery methods, both in terms of sensitivity and specificity. In particular, integrating different signatures such as read depth, read pair and split read, in the correct way, can improve the SV discover methods. Understanding the genotype-phenotype relation is a major problem of the field, and a complete study of variants in genomic regions is a crucial part this study. Natural future directions would be to study the contribution of genetic structural variations to phenotypes.

Appendix A

Maximum Parsimony SV with Conflict Resolution

The possibility of multiple mapping locations for each paired-end read raised the issue of which structural variations suggested by the maximal valid clusters are correct (i.e. a paired-end read that has multiple mappings can suggest at most one structural variation). We discussed this problem in details for different types of structural variation events in this thesis.

Note that the original Maximum Parsimony Structural Variation (MPSV) problem [55] has no limit on the number of SV predictions which occur in the same loci of the genome. A considerable amount of the predicated calls overlap with each other and a final post-processing heuristic to filter some of those overlapping predicted SVs was given (see the result Section of [55]).

As part of our contribution in a paper [57], we mathematically formulated these "conflicts" and modeled the structural variation discovery problem as a *novel* combinatorial optimization problem. In what follows, we present the Maximum Parsimony Structural Variation with Conflict Resolution problem and our solution for it.

A.1 Conflicting SV clusters in haploid and diploid genome sequences

We motivate the "conflict resolution" for structural variation using a simple example. Given paired-end reads from a haploid genome, a structural variation algorithm (such as VariationHunter, MoDil or BreakDancer) can construct two sets of paired-end clusters which support two independent deletions overlapping significantly as shown in Figure A.1. Since we assumed the genome was haploid,

both of the variations as shown in Figure A.1 cannot be correct simultaneously.

We will first formalize MPSV-CR for both haploid and diploid genomes and then will analyze the complexity of the MPSV-CR problem. Finally, we provide a heuristic to find a solution to MPSV-CR. We denoted this solution as VariationHunter-CR.

Assuming only a single haploid of a genome is sequenced, two valid clusters $Vclu_1$ and $Vclu_2$ of paired-end reads are *conflicting* if and only if there exists a potential scenario of structural variations suggested by the two valid clusters, such that existence of one of the events makes the other cluster not possible (similar to the case shown in Figure A.1). In [57], we presented the set of rules which determine if two valid clusters are conflicting in a haploid genome or not. We refer the reader to [57] for more details.

With the rules for two clusters being in conflict with each other or not in a haploid genome, we can model the conflict representation of all the clusters using a graph, denoted as *conflict graph*. Each cluster is a node, and there exists an edge between every two node, if and only if the two clusters represented by the two nodes are in conflict with each other. It is not hard to see that a valid set of SVs (that is a subset of node/clusters) is a set of nodes/clusters in which there are no two nodes in the subset which are connected by an edge. In another words the valid solution (without any conflicts) is an independent set of the conflict graph.

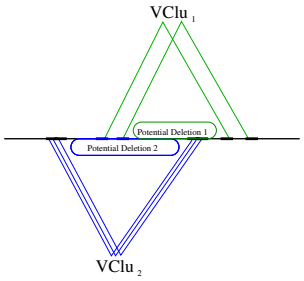


Figure A.1: In this figure, two valid clusters $Vclu_1$ and $Vclu_2$ are in conflict in a haploid genome.

One can easily generalize the definition of conflicting clusters to *diploid* genome sequences. Let $Vclu_1$ and $Vclu_2$ be two conflicting clusters in a haploid genome. However, providing the genome is diploid, both $Vclu_1$ and $Vclu_2$ can occur but in different haplotypes. Now consider a third SV cluster, $Vclu_3$, which conflicts with both $Vclu_1$ and $Vclu_2$. It is clear based on the pigeon hole principle that $Vclu_1$, $Vclu_2$, and $Vclu_3$ cannot occur in a diploid genome at the same time. In other words, the presence of three different SV clusters which are pairwise conflicting in a haploid genome

[1] will be a conflict in a diploid genome. In addition the concept of the conflict graphs for haploid genomes will be changed with haploid hypergraphs in diploid genomes. Where each hyperedge connects three nodes, if these three nodes are in conflict with each other, we denote this as a *conflict hypergraph*.

Now, we define the Maximum Parsimony Structural Variation with Conflict Resolution (MPSV-CR) problem which does not only ask to minimize the total number of implied structural variants but also guaranteesno pairwise conflicting triplet of the implied SVs exist.

Note that this new version of Maximum Parsimony Structural Variation problem does not select any conflicting structural variations and thus may not always be able to assign every paired-end read to a particular SV. Thus, the optimization function for MPSV-CR not only should try to minimize the number of SVs predicted, but also maximize the number of paired-ends that can be assigned to a location in genome.

In what follows we present the concept of conflicting SV clusters in more detail and give a formal definition of the MPSV-CR problem.

A.2 Formal definition of the MPSV-CR problem

In this section, we formally define the MPSV-CR problem. Let $MC = \{VClu_1, VClu_2, \cdots, VClu_n\}$ be the set of SV clusters and $R = \{pe_1, pe_2, \cdots, pe_n\}$ be the collection of discordant paired-end reads. These discordant end-pairs can have multiple locations in genome represented by $Align(pe_i) = \{a_1pe_i, a_2pe_i, \cdots, a_jpe_i\}$.

In order to formulate the constraints, we define the conflict hypergraph CG as a hypergraph with vertex set $V(CG) = MC$ and a hyperedge set $E(CG)$ as following: Among every *three* distinct SV clusters which are pair-wise conflicting, there exists a hyperedge in $E(CG)$:

$$\begin{aligned} E(CG) \;=\; & \{(VClu_i, VClu_j, VClu_k) \mid VClu_i, VClu_j, \\ & VClu_k \text{ are pairwise conflicting}\} \end{aligned} \tag{A.1}$$

We call a subset $SC \subset MC$ *satisfiable* under the constraint graph CG, if $\nexists e = (VClu_p, VClu_q, VClu_r) \in E(CG) : e \subseteq SC$. For each satisfiable subset SC and each paired-end read pe_i, we define the indicator variable $\delta(SC, pe_i)$ as follows:

[1] according to our definition of conflicts in a haploid genomes

$$\delta(SC, pe_i) = \begin{cases} 0 & \text{if } \exists SC_k \in SC \bigwedge \exists j : a_j pe_i \in SC_k \\ 1 & \text{otherwise} \end{cases}$$

Intuitively, $\delta(SC, pe_i)$ is the penalty for not assigning the pair-end read pe_i to a cluster in the satisfiable subset SC. The MPSV-CR problem aims to find the satisfiable set SC' such that $f(SC') = |SC'| + \sum_{pe \in R} \delta(SC', pe)$ is *minimized* (i.e. to find a trade-off between the number of SV clusters in a satisfiable set of SV clusters and the number of paired-end reads which do not assign to any of these SV clusters.). We will prove that MPSV-CR is NP-hard even if we have any positive weight on the cardinality of SC' and any positive penalty for unmapped reads (i.e. minimizing the function $g(SC') = k|SC'| + l \sum_{pe \in R} \delta(SC', pe)$ for some $k > 0$ and $l > 0$ is NP-hard.) When $l \geq k > 0$ (we denote this Case 1), minimizing $g(SC') = k|SC'| + l \sum_{pe \in R} \delta(SC', pe)$ is the same as minimizing $g(SC') = |SC'| + l' \sum_{pe \in R} \delta(SC', pe)$ where $l' = l/k$. When $k > l > 0$ (we denote this Case 2), minimizing $g(SC') = k|SC'| + l \sum_{pe \in R} \delta(SC', pe)$ is the same as minimizing $g(SC') = k'|SC'| + \sum_{pe \in R} \delta(SC', pe)$ where $k' = k/l$.

A.2.1 The MPSV-CR problem is NP-hard

In what follows, we prove that the MPSV-CR problem is NP-hard. We use a reduction from the minimum set cover problem for both cases. Given collection C of subsets of a finite set S ($|S| = n$), we would like to find a subset $C' \subseteq C$ with minimum cardinality such that every element in S belongs to at least one member of C'. We build an instance of MPSV-CR as followings:

Case 1 ($k = 1$ and $k \leq \ell$): For each element $S_i \in S$, there is a discordant paired-end read pe_i, and corresponding to each set $C_j \in C$ we have a cluster $VClu_j = C_j$. We define $R = S$, $V(CG) = V(G)$, and $E(CG) = \emptyset$. It is easy to see that if we have a set cover C' of size $\leq t$, we can pick a satisfiable set of clusters SC such that $g(SC) \leq t$. On the other hand, if we can pick a satisfiable set of clusters SC such that $g(SC) \leq t$ which includes x clusters and uncovers y discordant paired-end reads, we can have a corresponding solution $C'' \subseteq C$ for the set cover instance with $|C''| \leq x + y \leq t$ by choosing at most y more sets to cover y uncovered elements.

Case 2 ($\ell = 1$ and $\ell < k$): We denote $p = \lceil k \rceil$. For each element $S_i \in S$, there are p discordant paired-end reads $pe_i, pe_{i+n}, \ldots, pe_{i+np}$ and corresponding to each set $C_j \in C$ is a cluster $VClu_j = \{pe_k | S_{k \bmod n} \in C_j\}$. We define $R = S$, $V(CG) = V(G)$, and $E(CG) = \emptyset$ like in Case 1. If we have a set cover C' of size $\leq t$, we can pick a set of clusters SC such that $g(SC) = kt$.

When we can pick a satisfiable set of clusters SC such that $g(SC) \leq kt$ which includes x clusters with y uncovered discordant paired-end reads. By the construction, we have $g(SC) = kx + y$ and y uncovered discordant reads correspond to $y' = y/p$ elements in S. And since $k(x + y') \leq kx + py' \leq kx + y \leq kt$, we have $x + y' \leq t$. Thus, it is similar to Case 1 the collection C'' of x sets and at most additional y' sets to cover y' uncovered elements is a solution to the set cover instance with cardinality less than or equal to t.

A.3 An approximate solution to the MPSV-CR problem

In this part we present an approximation algorithm for MPSV-CR where $k = l = 1$ in the optimization function f or g. This solution has two steps:

- Assign as many discordant reads as possible to a *satisfiable* set of SV clusters; we define MR as the set of these pe_i's.
- Minimize the number of SV clusters that can cover all the reads in MR.

Let $maxMR$ be the maximum number of reads that can be assigned to a *satisfiable* set of SV clusters. We define $neighbors(VClu) = \{VClu' | \exists e \in E(CG), VClu, VClu' \in e\}$, $deg(VClu) = |neighbors(VClu)|$ and $\Delta = max\{deg(VClu) | VClu \in MC\}$ i.e the maximum degree of a vertex in the conflict graph CG. $Max\_Assigned\_Reads$ will be similar to the set cover algorithm in which at each iteration we choose the cluster that does not conflict with any other clusters and has maximum number of uncovered reads. We will prove that $Max\_Assigned\_Reads$ guarantees to return a set of assigned reads MR which has the cardinality at least $maxMR/(\Delta + 1)$ and also results in a $log(n)$ approximation of a minimum number of clusters that cover all the reads in MR.

Theorem 4. *$Max\_Assigned\_Reads$ returns set of covered reads MR with $|MR| \geq maxMR/(\Delta + 1)$.*

Proof. Letting m be the total number of reads, we show that $|MR| \geq m/\Delta$. Suppose that the algorithm terminates after k iterations. Denote MR_i, UR_i as the sets of reads that are covered and could not be covered for the first time at the i$-th$ iteration and $VClu_i$ as the cluster that gets picked. A read pe is called uncovered by a cluster $VClu_i$ if for each cluster $VClu'$: $ape \in VClu'$ and $VClu'$ could not be picked for the first time just after ith iteration, thus, $pe \in UR_i$. At ith iteration, the maximum number of reads that could be covered for all neighbors of $VClu_i$ is at most $\Delta|MR_i|$. Hence, the maximum number of reads that are uncovered for the first time is $|UR_i| \leq \Delta|MR_i|$.

Moreover, we have $\sum_{i=1}^{k}(UR_i + MR_i) = m$ and $|MR| = \sum_{i=1}^{k} MR_i$. Thus, it is followed that $m/(\Delta+1) \leq |MR|$. □

A.4 Structural Variant Prediction with VariationHunter-CR

Here we show that the calls predicted by VariationHunter with conflict resolution (VariationHunter-CR) has a lower false positive rate than the original VariationHunter [55], while retaining the same true positive rate as before.

We compare the deletion calls found using different algorithms including VariationHunter-CR on a Yoruba African donor (NA18507) recently sequenced with the Illumina Genome Analyzer [21] against validated deletions of the same individual reported in Kidd et al. [75]. We downloaded approximately 3.5 billion end sequences (~ 1.7 billion pairs) of length $36-41$bp and insert size 200bp from the NCBI Short Read Archive[2]. Similar to the pre-screening methodology which was used in [55], we removed any pairs of reads from consideration if either (or both) end sequences has average *phred* [38] quality value less than 20, or if either (or both) sequences contain more than 2 N characters. In total, ~ 1.3 billion reads were removed from this data set. We then mapped all the remaining ~ 2.2 billion end sequences to the human reference genome using *mrFAST* [7]. The average insert size was determined to be 209bp, where the standard deviation was 13.4bp.

We compare in Figure A.2 our new VariationHunter-CR algorithm against the original VariationHunter [55], the curated result published in [55] (some of VariationHunter calls were pruned using a post processing heuristic which removes the calls with less support if they are conflicting with each other), and BreakDancer [29].

We use a strict criteria to consider a validated call in Kidd et al [75] to be found by any method: First, the length of the call predicted should be at least 100bp or more. Second, it should be completely inside the call predicted by fosmid sequence (because the fosmid insert size is quite large in comparison to Illumina insert size, thus a call predicted by Illumina paired-end read is always inside a call predicted by fosmid sequence).

[2] ftp://ftp.ncbi.nih.gov/pub/TraceDB/ShortRead/SRA000271

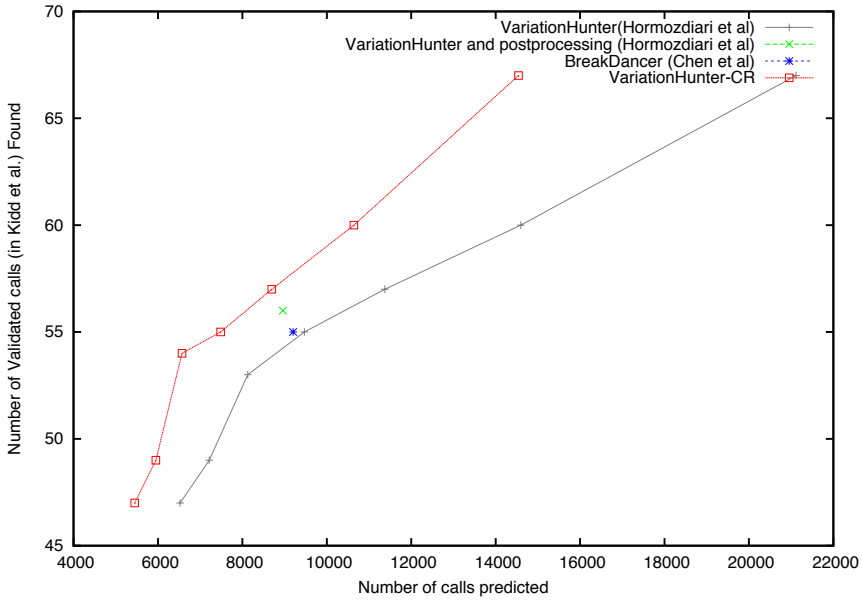


Figure A.2: Comparison for VariationHunter (Orange), curated result presented in [55] (Green) , BreakDancer (Blue) [29] and VariationHunter-CR (Red). The x-axis represents the number of deletion calls predicted by each method, and y-axis is the number of validated deletions form [75] which is found by each method. Thus, a result which is able to find more validated calls with less number of total calls is desirable. For VariationHunter and VariationHunter-CR we give the number of calls and number of validated deletions found for different support levels (number of paired-end read supporting them). As it can be seen VariationHunter-CR is given better results than VariationHunter for all the support levels (the red plot is always on top of the orange plot) and it also outperforms the result of BreakDancer published in [29]. In addition, VariationHunter-CR also outperformed MoDil's result published in [87], however because focus of MoDil is mainly finding medium and small size variations, we did not include it in this figure.

Bibliography

[1] 454 life sciences.

[2] Applied biosystems.

[3] Illumina, inc.

[4] Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 02 2001.

[5] The 1000GP. A map of human genome variation from population-scale sequencing. *Nature*, 467:1061–1073, 2010.

[6] Can Alkan, Bradley Coe, and Evan Eichler. Genome structural variation discovery and genotyping. *Nature Reviews Genetics*, 12(5):363–376, 03 2011.

[7] Can Alkan, Jeffrey M. Kidd, Tomas Marques-Bonet, Gozde Aksay, Francesca Antonacci, Fereydoun Hormozdiari, Jacob O. Kitzman, Carl Baker, Maika Malig, Onur Mutlu, S. Cenk Sahinalp, Richard A. Gibbs, and Evan E. Eichler. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature Genetics*, 41(10):1061–1067, 2009.

[8] Can Alkan, Saba Sajjadian, and Evan E Eichler. Limitations of next-generation genome sequence assembly. *Nat Meth*, 8(1):61–65, 01 2011.

[9] N. Alon and S. Gutner. Balanced families of perfect hash functions and their applications. *Proc. ICALP*, pages 435–446, 2007.

[10] Noga Alon, Phuong Dao, Iman Hajirasouliha, Fereydoun Hormozdiari, and S. Cenk Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 24(13):i241–i249, 2008.

[11] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[12] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[13] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *ISAAC '02: Proceedings of the 13th International Symposium on Algorithms and Computation*, pages 453–464, 2002.

[14] J. A. Bailey, L. Giu, and E. E. Eichler. An Alu transposition model for the origin and expansion of human segmental duplications. *Am J Hum Genet*, 73:823–34, 2003.

[15] Michael J. Bamshad, Stephen Wooding, W. Scott Watkins, Christopher T. Ostler, Mark A. Batzer, and Lynn B. Jorde. Human population genetic structure and inference of group membership. *The American Journal of Human Genetics*, 72(3):578–589, 3 2003.

[16] Ali Bashir, Stanislav Volik, Colin Collins, Vineet Bafna, and Benjamin J Raphael. Evaluation of paired-end sequencing strategies for detection of genome rearrangements in cancer. *PLoS Comput Biol*, 4(4):e1000051, Apr 2008.

[17] M. A. Batzer and P. L. Deininger. Alu repeats and human genomic diversity. *Nat Rev Genet*, 3(5):370–9, 2002.

[18] G. Bebek, P. Berenbrink, C. Cooper, T. Friedetzky, J. Nadeau, and S. C. Sahinalp. The degree distribution of the generalized duplication model. *Theor. Comput. Sci.*, 369(1):239–249, 2006.

[19] Christine R. Beck, Pamela Collier, Catriona Macfarlane, Maika Malig, Jeffrey M. Kidd, Evan E. Eichler, Richard M. Badge, and John V. Moran. Line-1 retrotransposition activity in human genomes. *Cell*, 141(7):1159–1170, 6 2010.

[20] C. Bekpen, T. Marques-Bonet, C. Alkan, F. Antonacci, M. B. Leogrande, M. Ventura, J. M. Kidd, P. Siswara, J. C. Howard, and E. E. Eichler. Death and resurrection of the human IRGM gene. *PLoS Genet.*, 5:e1000403, Mar 2009.

[21] David R Bentley, Shankar Balasubramanian, Harold P Swerdlow, Geoffrey P Smith, John Milton, Clive G Brown, Kevin P Hall, Dirk J Evers, Colin L Barnes, Helen R Bignell, Jonathan M Boutell, Jason Bryant, Richard J Carter, R. Keira Cheetham, Anthony J Cox, Darren J Ellis, Michael R Flatbush, Niall A Gormley, Sean J Humphray, Leslie J Irving, Mirian S Karbelashvili, Scott M Kirk, Heng Li, Xiaohai Liu, Klaus S Maisinger, Lisa J Murray, Bojan Obradovic, Tobias Ost, Michael L Parkinson, Mark R Pratt, Isabelle M J Rasolonjatovo, Mark T Reed, Roberto Rigatti, Chiara Rodighiero, Mark T Ross, Andrea Sabot, Subramanian V Sankar, Aylwyn Scally, Gary P Schroth, Mark E Smith, Vincent P Smith, Anastassia Spiridou, Peta E Torrance, Svilen S Tzonev, Eric H Vermaas, Klaudia Walter, Xiaolin Wu, Lu Zhang, Mohammed D Alam, Carole Anastasi, Ify C Aniebo, David M D Bailey, Iain R Bancarz, Saibal Banerjee, Selena G Barbour, Primo A Baybayan, Vincent A Benoit, Kevin F Benson, Claire Bevis, Phillip J Black, Asha Boodhun, Joe S Brennan, John A Bridgham, Rob C Brown, Andrew A Brown, Dale H Buermann, Abass A Bundu, James C Burrows, Nigel P Carter, Nestor Castillo, Maria Chiara E Catenazzi, Simon Chang, R. Neil Cooley, Natasha R Crake, Olubunmi O Dada, Konstantinos D Diakoumakos, Belen Dominguez-Fernandez, David J Earnshaw, Ugonna C Egbujor, David W Elmore, Sergey S

Etchin, Mark R Ewan, Milan Fedurco, Louise J Fraser, Karin V Fuentes Fajardo, W. Scott Furey, David George, Kimberley J Gietzen, Colin P Goddard, George S Golda, Philip A Granieri, David E Green, David L Gustafson, Nancy F Hansen, Kevin Harnish, Christian D Haudenschild, Narinder I Heyer, Matthew M Hims, Johnny T Ho, Adrian M Horgan, Katya Hoschler, Steve Hurwitz, Denis V Ivanov, Maria Q Johnson, Terena James, T. A. Huw Jones, Gyoung-Dong Kang, Tzvetana H Kerelska, Alan D Kersey, Irina Khrebtukova, Alex P Kindwall, Zoya Kingsbury, Paula I Kokko-Gonzales, Anil Kumar, Marc A Laurent, Cynthia T Lawley, Sarah E Lee, Xavier Lee, Arnold K Liao, Jennifer A Loch, Mitch Lok, Shujun Luo, Radhika M Mammen, John W Martin, Patrick G McCauley, Paul McNitt, Parul Mehta, Keith W Moon, Joe W Mullens, Taksina Newington, Zemin Ning, Bee Ling Ng, Sonia M Novo, Michael J O'Neill, Mark A Osborne, Andrew Osnowski, Omead Ostadan, Lambros L Paraschos, Lea Pickering, Andrew C Pike, Alger C Pike, D. Chris Pinkard, Daniel P Pliskin, Joe Podhasky, Victor J Quijano, Come Raczy, Vicki H Rae, Stephen R Rawlings, Ana Chiva Rodriguez, Phyllida M Roe, John Rogers, Maria C Rogert Bacigalupo, Nikolai Romanov, Anthony Romieu, Rithy K Roth, Natalie J Rourke, Silke T Ruediger, Eli Rusman, Raquel M Sanches-Kuiper, Martin R Schenker, Josefina M Seoane, Richard J Shaw, Mitch K Shiver, Steven W Short, Ning L Sizto, Johannes P Sluis, Melanie A Smith, Jean Ernest Sohna Sohna, Eric J Spence, Kim Stevens, Neil Sutton, Lukasz Szajkowski, Carolyn L Tregidgo, Gerardo Turcatti, Stephanie Vandevondele, Yuli Verhovsky, Selene M Virk, Suzanne Wakelin, Gregory C Walcott, Jingwen Wang, Graham J Worsley, Juying Yan, Ling Yau, Mike Zuerlein, Jane Rogers, James C Mullikin, Matthew E Hurles, Nick J McCooke, John S West, Frank L Oaks, Peter L Lundberg, David Klenerman, Richard Durbin, and Anthony J Smith. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218):53–59, Nov 2008.

[22] P. Bille. Tree edit distance, alignment and inclusion. Technical Report TR-2003-23, IT University of Copenhagen, 2003.

[23] Inanç Birol, Shaun D. Jackman, Cydney B. Nielsen, Jenny Q. Qian, Richard Varhol, Greg Stazyk, Ryan D. Morin, Yongjun Zhao, Martin Hirst, Jacqueline E. Schein, Doug E. Horsman, Joseph M. Connors, Randy D. Gascoyne, Marco A. Marra, and Steven J. M. Jones. De novo transcriptome assembly with abyss. *Bioinformatics*, 25(21):2872–2877, 2009.

[24] Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001.

[25] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A. Shlyakhter, Matthew K. Belmonte, Eric S. Lander, Chad Nusbaum, and David B. Jaffe. Allpaths: De novo assembly of whole-genome shotgun microreads. *Genome Research*, 18(5):810–820, 2008.

[26] Marion L Carroll, Astrid M Roy-Engel, Son V Nguyen, Abdel-Halim Salem, Erika Vogel, Bethaney Vincent, Jeremy Myers, Zahid Ahmad, Lan Nguyen, Mimi Sammarco, W. Scott Watkins, Jurgen Henke, Wojciech Makalowski, Lynn B Jorde, Prescott L Deininger, and Mark A Batzer. Large-scale analysis of the alu ya5 and yb8 subfamilies and their contribution to human genomic diversity. *Journal of Molecular Biology*, 311(1):17–40, 8 2001.

[27] Mark Chaisson, Pavel Pevzner, and Haixu Tang. Fragment assembly with short reads. *Bioinformatics*, 20(13):2067–2074, Sep 2004.

[28] Mark J Chaisson and Pavel A Pevzner. Short read fragment assembly of bacterial genomes. *Genome Res*, 18(2):324–330, Feb 2008.

[29] Ken Chen, John Wallis, Michael McLellan, David Larson, Joelle Kalicki, Craig Pohl, Sean McGrath, Michael Wendl, Qunyuan Zhang, Devin Locke, Xiaoqi Shi, Robert Fulton, Timothy Ley, Richard Wilson, Li Ding, and Elaine Mardis. Breakdancer: an algorithm for high-resolution mapping of genomic structural variation. *Nature Methods.*, 6:677 – 681, 2009.

[30] Michael James Clark, Nils Homer, Brian D. O'Connor, Zugen Chen, Ascia Eskin, Hane Lee, Barry Merriman, and Stanley F. Nelson. U87mg decoded: The genomic sequence of a cytogenetically aberrant human cancer cell line. *PLoS Genet*, 6(1):e1000832, 01 2010.

[31] International Cancer Genomes Consortium. International network of cancer genome projects. *Nature*, 464:993–998, 2010.

[32] Richard Cordaux, Deepa Srikanta, Jungnam Lee, Mark Stoneking, and Mark A. Batzer. In search of polymorphic alu insertions with restricted geographic distributions. *Genomics*, 90(1):154–158, 7 2007.

[33] Richard Cordauxa, Dale J. Hedgesa, Scott W. Herkea, and Mark A. Batzer. Estimating the retrotransposition rate of human alu elements. *Gene*, 373:134–137, 2006.

[34] Prescott L. Deininger, Douglas J. Jolly, Carol M. Rubin, Theodore Friedmann, and Carl W. Schmid. Base sequence studies of 300 nucleotide renatured repeated human dna clones. *Journal of Molecular Biology*, 151(1):17–33, 9 1981.

[35] Banu Dost, Tomer Shlomi, Nitin Gupta 0002, Eytan Ruppin, Vineet Bafna, and Roded Sharan. Qnet: A tool for querying protein interaction networks. In *RECOMB*, pages 1–15, 2007.

[36] Dent A. Earl, Keith Bradnam, John St. John, Aaron Darling, Dawei Lin, Joseph Faas, Hung On Ken Yu, Buffalo Vince, Daniel R. Zerbino, Mark Diekhans, Ngan Nguyen, Pramila Nuwantha, Ariyaratne Wing-Kin Sung, Zemin Ning, Matthias Haimel, Jared T. Simpson, Nuno A. Fronseca, İnanç Birol, T. Roderick Docking, Isaac Y. Ho, Daniel S Rokhsar, Rayan Chikhi, Dominique Lavenier, Guillaume Chapuis, Delphine Naquin, Nicolas Maillet, Michael C. Schatz, David R. Kelly, Adam M. Phillippy, Sergey Koren, Shiaw-Pyng Yang, Wei Wu, Wen-Chi Chou, Anuj Srivastava, Timothy I. Shaw, J. Graham Ruby, Peter Skewes-Cox, Miguel Betegon, Michelle T. Dimon, Victor Solovyev, Petr Kosarev, Denis Vorobyev, Ricardo Ramirez-Gonzalez, Richard Leggett, Dan MacLean, Fangfang Xia, Ruibang Luo, Zhenyu L, Yinlong Xie, Binghang Liu, Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J. Ribeiro, Shuangye Yin, Ted Sharpe, Giles Hall, Paul J. Kersey, Richard Durbin, Shaun D. Jackman, Jarrod A. Chapman, Xiaoqiu Huang, Joseph L. DeRisi, Mario Caccamo, Yingrui Li, David B. Jaffe, Richard Green, David Haussler, Ian Korf, and Benedict Paten. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research*, 2011.

[37] Adam D. Ewing and Haig H. Kazazian. High-throughput sequencing reveals extensive variation in human-specific l1 content in individual human genomes. *Genome Research*, 20(9):1262–1270, 2010.

[38] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Res*, 8(3):186–94, 1998.

[39] L. Feuk, A. R. Carson, and S. W. Scherer. Structural variation in the human genome. *Nat Rev Genet*, 7(2):85–97, 2006.

[40] Daya Ram Gaur, Ramesh Krishnamurti, and Rajeev Kohli. The capacitated max ¡i¿k¡/i¿-cut problem. *Math. Program.*, 115(1):65–72, May 2008.

[41] J. Gertz, G. Elfond, A. Shustrova, M. Weisinger, M. Pellegrini, S. Cokus, and B. Rothschild. Inferring protein interactions from phylogenetic distance matrices. *Bioinformatics*, 19(16):2039–2045, 2003.

[42] P. Green. cross-match. at http://www.phrap.org.

[43] Richard E. Green, Johannes Krause, Adrian W. Briggs, Tomislav Maricic, Udo Stenzel, Martin Kircher, Nick Patterson, Heng Li, Weiwei Zhai, Markus Hsi-Yang Fritz, Nancy F. Hansen, Eric Y. Durand, Anna-Sapfo Malaspinas, Jeffrey D. Jensen, Tomas Marques-Bonet, Can Alkan, Kay Prüfer, Matthias Meyer, Hernán A. Burbano, Jeffrey M. Good, Rigo Schultz, Ayinuer Aximu-Petri, Anne Butthof, Barbara Höber, Barbara Höffner, Madlen Siegemund, Antje Weihmann, Chad Nusbaum, Eric S. Lander, Carsten Russ, Nathaniel Novod, Jason Affourtit, Michael Egholm, Christine Verna, Pavao Rudan, Dejana Brajkovic, Željko Kucan, Ivan Gušic, Vladimir B. Doronichev, Liubov V. Golovanova, Carles Lalueza-Fox, Marco de la Rasilla, Javier Fortea, Antonio Rosas, Ralf W. Schmitz, Philip L. F. Johnson, Evan E. Eichler, Daniel Falush, Ewan Birney, James C. Mullikin, Montgomery Slatkin, Rasmus Nielsen, Janet Kelso, Michael Lachmann, David Reich, and Svante Pääbo. A draft sequence of the neandertal genome. *Science*, 328(5979):710–722, 2010.

[44] Joshua A. Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *RECOMB*, pages 92–106, 2007.

[45] U. I. Gupta, T. D. Lee, and J. Y. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12:459–467, 1982.

[46] F. Hach, F. Hormozdiari, C. Alkan, I. Birol, E. E. Eichler, and S.C. Sahinalp. Cache oblivious algorithms for high throughput read mapping. *Nature Meth*, 2010.

[47] I. Hajirasouliha, F. Hormozdiari, S. C. Sahinalp, and I. Birol. Optimal pooling for genome re-sequencing with ultra-high-throughput short-read technologies. *Bioinformatics (Oxford, England)*, 24(13), July 2008.

[48] Iman Hajirasouliha, Fereydoun Hormozdiari, Can Alkan, Jeffrey M. Kidd, Inanc Birol, Evan E. Eichler, and S. Cenk Sahinalp. Detection and characterization of novel sequence insertions using paired-end next-generation sequencing. *Bioinformatics*, 26(10):1277–1283, 2010.

[49] Iman Hajirasouliha, Alexander Schönhuth, David de Juan, Alfonso Valencia, and S. Cenk Sahinalp. Mirroring co-evolving trees in the light of their topologies. *Bioinformatics*, 28(9):1202–1208, 2012.

[50] J. Han, D. Dupuy, N. Bertin, M. Cusick, and M. Vidal. Effect of sampling on topology predictions of protein-protein interaction networks. *Nature Biotech*, 23:839–844, 2005.

[51] Robert E Handsaker, Joshua M Korn, James Nemesh, and Steven A McCarroll. Discovery and genotyping of genome structural polymorphism by sequencing on a population scale. *Nat Genet*, 43(3):269–276, 03 2011.

[52] Edward Hollox, Ulrike Huffmeier, Patrick Zeeuwen, Raquel Palla, Jesús Lascorz, Diana Rodijk-Olthuis, Peter Kerkhof, Heiko Traupe, Gys Jongh, Martin Heijer, André Reis, John Armour, and Joost Schalkwijk. Psoriasis is associated with increased bold beta-defensin genomic copy number. *Nature Genetics*, 40:23–25, 2008.

[53] F. Hormozdiari, P. Berenbrink, N. Przulj, , and S.C. Sahinalp. Not all scale-free networks are born equal: The role of the seed graph in ppi network evolution. *PLoS Comput Biol*, 3(7), 2007.

[54] Farhad Hormozdiari, Faraz Hach, S. Cenk Sahinalp, Evan E. Eichler, and Can Alkan. Sensitive and fast mapping of di-base encoded reads. *Bioinformatics*, 27(14):1915–1921, 2011.

[55] Fereydoun Hormozdiari, Can Alkan, Evan E. Eichler, and S. Cenk Sahinalp. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Recomb 2009/Genome Research*, 19(7):1270–1278, Jul 2009.

[56] Fereydoun Hormozdiari, Can Alkan, Mario Ventura, Iman Hajirasouliha, Maika Malig, Faraz Hach, Deniz Yorukoglu, Phuong Dao, Marzieh Bakhshi, S. Cenk Sahinalp, and Evan E. Eichler. Alu repeat discovery and characterization within human genomes. *Genome Research*, Dec 2010.

[57] Fereydoun Hormozdiari, Iman Hajirasouliha, Phuong Dao, Faraz Hach, Deniz Yorukoglu, Can Alkan, Evan E. Eichler, and S. Cenk Sahinalp. Next-generation variationhunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*, 26(12):i350–i357, 2010.

[58] Fereydoun Hormozdiari, Iman Hajirasouliha, Andrew McPherson, Evan E. Eichler, and S. Cenk Sahinalp. Simultaneous structural variation discovery among multiple paired-end sequenced genomes. *Genome Research*, 21(12):2203–2212, December 2011.

[59] Catherine M. Houck, Frank P. Rinehart, and Carl W. Schmid. A ubiquitous family of repeated dna sequences in the human genome. *Journal of Molecular Biology*, 132(3):289–306, 8 1979.

[60] Cheng Ran Lisa Huang, Anna M. Schneider, Yunqi Lu, Tejasvi Niranjan, Peilin Shen, Matoya A. Robinson, Jared P. Steranka, David Valle, Curt I. Civin, Tao Wang, Sarah J. Wheelan, Hongkai Ji, Jef D. Boeke, and Kathleen H. Burns. Mobile interspersed repeats are major structural variants in the human genome. *Cell*, 141(7):1171–1182, 6 2010.

[61] IHGSC. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–45, 2004.

[62] IHMC. The international HapMap project. *Nature*, 426(6968):789–796, Dec 2003.

[63] IHMC. A haplotype map of the human genome. *Nature*, 437(7063):1299–320, 2005.

[64] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nature genetics*, 44(2):226–232, February 2012.

[65] Rebecca C. Iskow, Michael T. McCabe, Ryan E. Mills, Spencer Torene, W. Stephen Pittard, Andrew F. Neuwald, Erwin G. Van Meir, Paula M. Vertino, and Scott E. Devine. Natural mutagenesis of human genomes by endogenous retrotransposons. *Cell*, 141(7):1253–1261, 6 2010.

[66] J.M.G. Izarzugaza, D. Juan, C. Pons, F. Pazos, and A. Valencia. Enhancing the prediction of protein pairings between interacting families using orthology information. *BMC Bioinformatics*, 9:35, 2008.

[67] Wang J, Song L, Grover D, Azrak S, Batzer MA, and Liang P. dbrip: a highly integrated database of retrotransposon insertion polymorphisms in humans. *Hum. Mutt.*, 27:323–329, 2006.

[68] R. Jothi, M.G. Kann, and T.M. Przytycka. Predicting protein-protein interaction by searching evolutionary tree automorphism space. *Bioinformatics*, 21(Suppl.1):i241–i250, 2005.

[69] J. Jurka, O. Kohany, A. Pavlicek, V. V. Kapitonov, and M. V. Jurka. Duplication, coclustering, and selection of human Alu retrotransposons. *Proc Natl Acad Sci U S A*, 101(5):1268–72, 2004.

[70] Emre Karakoc, Can Alkan, Brian J O'Roak, Megan Y Dennis, Laura Vives, Kenneth Mark, Mark J Rieder, Debbie A Nickerson, and Evan E Eichler. Detection of structural variants and indels within exome data. *Nat Meth*, advance online publication:–, 12 2011.

[71] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85—103, 1972.

[72] Richard M. Karp and Michael Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64, 1983.

[73] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics.*, Jul 22; 20(11):1746–58, 2004.

[74] W. James Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4):656–664, Apr 2002.

[75] Jeffrey M. Kidd, Gregory M. Cooper, William F. Donahue, Hillary S. Hayden, Nick Sampas, Tina Graves, Nancy Hansen, Brian Teague, Can Alkan, Francesca Antonacci, Eric Haugen, Troy Zerr, N. Alice Yamada, Peter Tsang, Tera L. Newman, Eray Tüzün, Ze Cheng, Heather M. Ebling, Nadeem Tusneem, Robert David, Will Gillett, Karen A. Phelps, Molly Weaver, David Saranga, Adrianne Brand, Wei Tao, Erik Gustafson, Kevin McKernan, Lin Chen, Maika Malig, Joshua D. Smith, Joshua M. Korn, Steven A. McCarroll, David A. Altshuler, Daniel A. Peiffer, Michael Dorschner, John Stamatoyannopoulos, David Schwartz, Deborah A. Nickerson, James C. Mullikin, Richard K. Wilson, Laurakay Bruhn, Maynard V. Olson, Rajinder Kaul, Douglas R. Smith, and Evan E. Eichler. Mapping and sequencing of structural variation from eight human genomes. *Nature*, 453(7191):56–64, May 2008.

[76] Jeffrey M. Kidd, Tina Graves, Tera L. Newman, Robert Fulton, Hillary S. Hayden, Maika Malig, Joelle Kallicki, Rajinder Kaul, Richard K. Wilson, and Evan E. Eichler. A human genome structural variation sequencing resource reveals insights into mutational mechanisms. *Cell*, 143(5):837–847, 11 2010.

[77] Jeffrey M Kidd, Nick Sampas, Francesca Antonacci, Tina Graves, Robert Fulton, Hillary S Hayden, Can Alkan, Maika Malig, Mario Ventura, Giuliana Giannuzzi, Joelle Kallicki, Paige Anderson, Anya Tsalenko, N Alice Yamada, Peter Tsang, Rajinder Kaul, Richard K Wilson, Laurakay Bruhn, and Evan E Eichler. Characterization of missing human genome sequences and copy-number polymorphic insertions. *Nat Meth*, 7(5):365–371, 05 2010.

[78] Jong-Il Kim, Young Seok Ju, Hansoo Park, Sheehyun Kim, Seonwook Lee, Jae-Hyuk Yi, Joann Mudge, Neil A. Miller, Dongwan Hong, Callum J. Bell, Hye-Sun Kim, In-Soon Chung, Woo-Chung Lee, Ji-Sun Lee, Seung-Hyun Seo, Ji-Young Yun, Hyun Nyun Woo, Heewook Lee, Dongwhan Suh, Seungbok Lee, Hyun-Jin Kim, Maryam Yavartanoo, Minhye Kwak, Ying Zheng, Mi Kyeong Lee, Hyunjun Park, Jeong Yeon Kim, Omer Gokcumen, Ryan E. Mills, Alexander Wait Zaranek, Joseph Thakuria, Xiaodi Wu, Ryan W. Kim, Jim J. Huntley, Shujun Luo, Gary P. Schroth, Thomas D. Wu, HyeRan Kim, Kap-Seok Yang, Woong-Yang Park, Hyungtae Kim, George M. Church, Charles Lee, Stephen F. Kingsmore, and Jeong-Sun Seo. A highly annotated whole-genome sequence of a korean individual. *Nature*, 460(7258):1011–1015, 08 2009.

[79] Jacob O Kitzman, Alexandra P MacKenzie, Andrew Adey, Joseph B Hiatt, Rupali P Patwardhan, Peter H Sudmant, Sarah B Ng, Can Alkan, Ruolan Qiu, Evan E Eichler, and Jay Shendure. Haplotype-resolved genome sequencing of a gujarati indian individual. *Nat Biotech*, 29(5):459–459, 05 2011.

[80] J. O. Korbel, A. Abyzov, X. J. Mu, N. Carriero, P. Cayting, Z. Zhang, M. Snyder, and M. B. Gerstein. PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data. *Genome Biol.*, 10:R23, Feb 2009.

[81] Jan O Korbel, Alexander Eckehart Urban, Jason P Affourtit, Brian Godwin, Fabian Grubert, Jan Fredrik Simons, Philip M Kim, Dean Palejev, Nicholas J Carriero, Lei Du, Bruce E Taillon, Zhoutao Chen, Andrea Tanzer, A. C Eugenia Saunders, Jianxiang Chi, Fengtang Yang, Nigel P Carter, Matthew E Hurles, Sherman M Weissman, Timothy T Harkins, Mark B Gerstein, Michael Egholm, and Michael Snyder. Paired-end mapping reveals extensive structural variation in the human genome. *Science*, 318(5849):420–426, Oct 2007.

[82] Martin Krzywinski, Ian Bosdet, Carrie Mathewson, Natasja Wye, Jay Brebner, Readman Chiu, Richard Corbett, Matthew Field, Darlene Lee, Trevor Pugh, Stas Volik, Asim Siddiqui, Steven Jones, Jacquie Schein, Collin Collins, and Marco Marra. A bac clone fingerprinting approach to the detection of human genome rearrangements. *Genome Biology*, 8, 10 2007.

[83] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with bowtie 2. *Nature Methods*, 9(4):357–359, March 2012.

[84] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology*, 10(3):R25+, 2009.

[85] RJ. Leary, I. Kinde, F. Diehl, K. Schmidt, C. Clouser, C. Duncan, A. Antipova, C. Lee, K. McKernan, FM. De La Vega, KW. Kinzler, B. Vogelstein, LA Jr. Diaz, and VE. Velculescu. Development of personalized tumor biomarkers using massively parallel sequencing. *Sci. Transl. Med.*, 2:p20ra14, 2010.

[86] Seunghak Lee, Elango Cheran, and Michael Brudno. A robust framework for detecting structural variations in a genome. *Bioinformatics*, 24(13):i59–i67, Jul 2008.

[87] Seunghak Lee, Fereydoun Hormozdiari, Can Alkan, and Michael Brudno. Modil: detecting small indels from clone-end sequencing with mixtures of distributions. *Nature Methods*, 6:473 – 474, 2009.

[88] Seunghak Lee, Eric Xing, and Michael Brudno. Mogul: Detecting common insertions and deletions in a population. *Recomb/LNCS*, 6044:357–368, 2010.

[89] Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, Yuan Lin, Jeffrey R MacDonald, Andy Wing Chun Pang, Mary Shago, Timothy B Stockwell, Alexia Tsiamouri, Vineet Bafna, Vikas Bansal, Saul A Kravitz, Dana A Busam, Karen Y Beeson, Tina C McIntosh, Karin A Remington, Josep F Abril, John Gill, Jon Borman, Yu-Hui Rogers, Marvin E Frazier, Stephen W Scherer, Robert L Strausberg, and J. Craig Venter. The diploid genome sequence of an individual human. *PLoS Biol*, 5(10):e254, Sep 2007.

[90] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, 26(5):589–595, March 2010.

[91] Heng Li, Jue Ruan, and Richard Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Res*, 18(11):1851–1858, Nov 2008.

[92] Ruiqiang Li, Hongmei Zhu, Jue Ruan, Wubin Qian, Xiaodong Fang, Zhongbin Shi, Yingrui Li, Shengting Li, Gao Shan, Karsten Kristiansen, Songgang Li, Huanming Yang, Jian Wang, and Jun Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, February 2010.

[93] George E. Liu, Can Alkan, Lu Jiang, Shaying Zhao, and Evan E. Eichler. Comparative analysis of alu repeats in primate genomes. *Genome Research*, 19(5):876–885, 2009.

[94] S.C. Lovell and D.L. Robertson. An integrated view of molecular co-evolution in protein-protein interactions. *Mol. Biol. Evol.*, 27(11):2567–2575, 2010.

[95] Elaine R Mardis. Cancer genomics identifies determinants of tumor biology. *Genome Biol.*, 11(5):211, May 2010.

[96] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, J.S. Bader, L.A. Bemben, J. Berka, M.S. Braverman, Y.J. Chen, Z. Chen, S.B. Dewell, L. Du, J.M. Fierro, X.V. Gomes, B.C. Godwin, W. He, S. Helgesen, C.H. Ho, G.P. Irzyk, S.C. Jando, M.L. Alenquer, T.P. Jarvie, K.B. Jirage, J.B. Kim, J.R. Knight, J.R. Lanza, J.H. Leamon, S.M. Lefkowitz, M. Lei, J. Li, K.L. Lohman, H. Lu, V.B. Makhijani, K.E. McDade, M.P. McKenna, E.W. Myers, E. Nickerson, J.R. Nobile, R. Plant, B.P. Puc, M.T. Ronan, G.T. Roth, G.J. Sarkis, J.F. Simons, J.W. Simpson, M. Srinivasan, K.R. Tartaro, A. Tomasz, K.A. Vogt, G.A. Volkmer, S.H. Wang, Y. Wang, M.P. Weiner, P. Yu, R.F. Begley, and J.M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, Sep 15 2005.

[97] Carrie A. Mathewson, Jacqueline E. Schein, and Marco A. Marra. *Large-Scale BAC Clone Restriction Digest Fingerprinting*. John Wiley & Sons, Inc., 2007.

[98] 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.*, Aug 2008.

[99] Steven A. McCarroll. Copy number variation and human genome maps. *Nature Genetics*, 42(5):365–366, May 2010.

[100] Edward M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, April 1976.

[101] Kevin Judd McKernan, Heather E. Peckham, Gina L. Costa, Stephen F. McLaughlin, Yutao Fu, Eric F. Tsung, Christopher R. Clouser, Cisyla Duncan, Jeffrey K. Ichikawa, Clarence C.

Lee, Zheng Zhang, Swati S. Ranade, Eileen T. Dimalanta, Fiona C. Hyland, Tanya D. Sokolsky, Lei Zhang, Andrew Sheridan, Haoning Fu, Cynthia L. Hendrickson, Bin Li, Lev Kotler, Jeremy R. Stuart, Joel A. Malek, Jonathan M. Manning, Alena A. Antipova, Damon S. Perez, Michael P. Moore, Kathleen C. Hayashibara, Michael R. Lyons, Robert E. Beaudoin, Brittany E. Coleman, Michael W. Laptewicz, Adam E. Sannicandro, Michael D. Rhodes, Rajesh K. Gottimukkala, Shan Yang, Vineet Bafna, Ali Bashir, Andrew MacBride, Can Alkan, Jeffrey M. Kidd, Evan E. Eichler, Martin G. Reese, Francisco M. De La Vega, and Alan P. Blanchard. Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. *Genome Research*, 19(9):1527–1541, 2009.

[102] Andrew McPherson, Fereydoun Hormozdiari, Abdalnasser Zayed, Ryan Giuliany, Gavin Ha, Mark G. Sun, Malachi Griffith, Alireza Heravi Moussavi, Janine Senz, Nataliya Melnyk, Marina Pacheco, Marco A. Marra, Martin Hirst, Torsten O. Nielsen, S. Cenk Sahinalp, David Huntsman, and Sohrab P. Shah. deFuse: an algorithm for gene fusion discovery in tumor RNA-seq data. *PLoS computational biology*, 7(5):e1001138+, May 2011.

[103] Andrew McPherson, Chunxiao Wu, Iman Hajirasouliha, Fereydoun Hormozdiari, Faraz Hach, Anna Lapuk, Stanislav Volik, Sohrab Shah, Colin Collins, and S. Cenk Sahinalp. Comrad: detection of expressed rearrangements by integrated analysis of rna-seq and low coverage genome sequence data. *Bioinformatics*, 27(11):1481–1488, 2011.

[104] Andrew W. McPherson, Chunxiao Wu, Alexander Wyatt, Sohrab P. Shah, Colin Collins, and S. Cenk Sahinalp. nFuse: Discovery of complex genomic rearrangements in cancer using high-throughput sequencing. *Genome Research*, June 2012.

[105] P. Medvedev, M. Stanciu, and M. Brudno. Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods*, 6:13–20, Nov 2009.

[106] Paul Medvedev, Marc Fiume, Misko Dzamba, Tim Smith, and Michael Brudno. Detecting copy number variation with mated short reads. *Genome Research*, 20(11):1613–1622, 2010.

[107] R. E. Mills, E. A. Bennett, R. C. Iskow, and S. E. Devine. Which transposable elements are active in the human genome? *Trends Genet.*, 23:183–191, Apr 2007.

[108] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[109] James Munkres. Algorithms for the assignment and transportation problems. 5(1):32–38, March 1957.

[110] Alon N., Moshkovitz D., and Safra S. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms (ACM)*, 2:153–177, 2006.

[111] Hansoo Park, Jong-Il Kim, Young Seok Ju, Omer Gokcumen, Ryan E Mills, Sheehyun Kim, Seungbok Lee, Dongwhan Suh, Dongwan Hong, Hyunseok Peter Kang, Yun Joo Yoo, Jong-Yeon Shin, Hyun-Jin Kim, Maryam Yavartanoo, Young Wha Chang, Jung-Sook Ha, Wilson

Chong, Ga-Ram Hwang, Katayoon Darvishi, HyeRan Kim, Song Ju Yang, Kap-Seok Yang, Hyungtae Kim, Matthew E Hurles, Stephen W Scherer, Nigel P Carter, Chris Tyler-Smith, Charles Lee, and Jeong-Sun Seo. Discovery of common asian copy number variants using integrated high-resolution array cgh and massively parallel dna sequencing. *Nat Genet*, 42(5):400–405, 05 2010.

[112] F. Pazos and A. Valencia. Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering*, 14(9):609–614, 2001.

[113] F. Pazos and A. Valencia. Protein co-evolution, co-adaptation and interactions. *The EMBO Journal*, 27(20):2648–2655, 2008.

[114] M. Pellegrini, E.M. Marcotte, M.J. Thompson, D. Eisenberg, and T.O. Yates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA*, 96(8):4285–8, 1999.

[115] P. A. Pevzner, H. Tang, and M. S. Waterman. An eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci U S A*, 98(17):9748–53, 2001.

[116] R.Y. Pinter, O. Rokhlenko, D. Tsur, and M. Ziv-Ukelson. Approximate labelled subtree homeomorphism. *Journal of Discrete Algorithms*, 6:480–496, 2008.

[117] Alkes L. Price, Eleazar Eskin, and Pavel A. Pevzner. Whole-genome analysis of alu repeat elements reveals complex evolutionary history. *Genome Research*, 14(11):2245–2252, 2004.

[118] Natasa Przulj, Derek G. Corneil, and Igor Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[119] Aaron R Quinlan, Royden A Clark, Svetlana Sokolova, Mitchell L Leibowitz, Yujun Zhang, Matthew E Hurles, Joshua C Mell, and Ira M Hall. Genome-wide mapping and assembly of structural variant breakpoints in the mouse genome. *Genome Res*, Mar 2010.

[120] Raz R. and Safra S. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. *STOC*, pages 475–484, 1997.

[121] A.K. Ramani and E.M. Marcotte. Exploiting the co-evolution of interacting proteins to discover interaction specificity. *J. Mol. Biol.*, 327:273–284, 2003.

[122] Benjamin J Raphael, Stanislav Volik, Colin Collins, and Pavel A Pevzner. Reconstructing tumor genome architectures. *Bioinformatics*, 19 Suppl 2:ii162–ii171, Oct 2003.

[123] B. Raynal, M. Couprie, and V. Biri. Homeomorphic alignment of weighted trees. *Pattern Recognition*, 43:2937–2949, 2010.

[124] Mills RE, Walter K, Stewart C, Handsaker R, Chen K, Alkan C, Abyzov A, Yoon SC, Ye K, Cheetham RK, and et al. Mapping copy number variation by population-scale genome sequencing. *Nature*, 470:59–65, 2010.

[125] Adam Roberts, Harold Pimentel, Cole Trapnell, and Lior Pachter. Identification of novel transcripts in annotated genomes using rna-seq. *Bioinformatics*, 2011.

[126] Suleyman Cenk Sahinalp and Uzi Vishkin. Symmetry breaking for suffix tree construction. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 300–309, New York, NY, USA, 1994. ACM.

[127] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976.

[128] Abdel-Halim Salem, Gail E. Kilroy, W. Scott Watkins, Lynn B. Jorde, and Mark A. Batzer. Recently integrated alu elements and human genomic diversity. *Molecular Biology and Evolution*, 20(8):1349–1361, 2003.

[129] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12):5463–5467, Dec 1977.

[130] Carl W. Schmid and Prescott L. Deininger. Sequence organization of the human genome. *Cell*, 6(3):345–358, 11 1975.

[131] Stephan C. Schuster, Webb Miller, Aakrosh Ratan, Lynn P. Tomsho, Belinda Giardine, Lindsay R. Kasson, Robert S. Harris, Desiree C. Petersen, Fangqing Zhao, Ji Qi, Can Alkan, Jeffrey M. Kidd, Yazhou Sun, Daniela I. Drautz, Pascal Bouffard, Donna M. Muzny, Jeffrey G. Reid, Lynne V. Nazareth, Qingyu Wang, Richard Burhans, Cathy Riemer, Nicola E. Wittekindt, Priya Moorjani, Elizabeth A. Tindall, Charles G. Danko, Wee Siang Teo, Anne M. Buboltz, Zhenhai Zhang, Qianyi Ma, Arno Oosthuysen, Abraham W. Steenkamp, Hermann Oostuisen, Philippus Venter, John Gajewski, Yu Zhang, B. Franklin Pugh, Kateryna D. Makova, Anton Nekrutenko, Elaine R. Mardis, Nick Patterson, Tom H. Pringle, Francesca Chiaromonte, James C. Mullikin, Evan E. Eichler, Ross C. Hardison, Richard A. Gibbs, Timothy T. Harkins, and Vanessa M. Hayes. Complete khoisan and bantu genomes from southern africa. *Nature*, 463(7283):943–947, 02 2010.

[132] J. Scott, T. Ideker, R. M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *J Comput Biol*, 13(2):133–144, 2006.

[133] Tomer Shlomi, Daniel Segal, Eytan Ruppin, and Roded Sharan. Qpath: a method for querying pathways in a protein-protein interaction network. *BMC Bioinformatics*, 7:200, 2006.

[134] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J.M. Jones, and İnanç Birol. Abyss: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[135] Suzanne Sindi, Selim Onal, Luke Peng, Hsin T. Wu, and Benjamin Raphael. An integrative probabilistic model for identification of structural variation in sequencing data. *Genome Biology*, 13(3):R22+, 2012.

[136] Suzanne S. Sindi, Elena Helman, Ali Bashir, and Benjamin J. Raphael. A geometric approach for classification and comparison of structural variants. *Bioinformatics*, 25(12), 2009.

[137] A. F. A. Smit, R. Hubley, and P. Green. Repeatmasker. at http://www.repeatmasker.org, 2010.

[138] P. Stankiewicz and J. R. Lupski. Structural variation in the human genome and its role in disease. *Annual Review of Medicine*, 61:437–455, 2010.

[139] Peter H. Sudmant, Jacob O. Kitzman, Francesca Antonacci, Can Alkan, Maika Malig, Anya Tsalenko, Nick Sampas, Laurakay Bruhn, Jay Shendure, 1000 Genomes Project, and Evan E. Eichler. Diversity of human copy number variation and multicopy genes. *Science (New York, N.Y.)*, 330(6004):641–646, October 2010.

[140] Andreas Sundquist, Mostafa Ronaghi, Haixu Tang, Pavel Pevzner, and Serafim Batzoglou. Whole-genome sequencing and assembly with high-throughput, short-read technologies. *PLoS ONE*, 2(5), 05 2007.

[141] K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26:422–433, 1979.

[142] J.D. Thompson, D.G. Higgins, and T.J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignments through sequence weighting, position specific gap penalties and weight matrix choice. *Nucl. Acids Res*, 18(22):4673–4680, 1994.

[143] E.R.M. Tillier, L. Biro, L. Ginny, and D. Tillo. Codep: Maximizing co-evolutionary interdependencies to discover interacting proteins. *Proteins: Structure, Function and Bioinformatics*, 63:822–831, 2006.

[144] E.R.M. Tillier and R.L. Charlebois. The human protein coevolution network. *Genome Research*, 19:1861–1871, 2009.

[145] Cole Trapnell, Lior Pachter, and Steven L. Salzberg. TopHat: discovering splice junctions with RNA-seq. *Bioinformatics*, 25(9):1105–1111, May 2009.

[146] Todd J. Treangen and Steven L. Salzberg. Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nat Rev Genet*, 13(1):36–46, 01 2012.

[147] E. Tuzun, A. J. Sharp, J. A. Bailey, R. Kaul, V. A. Morrison, L. M. Pertz, E. Haugen, H. Hayden, D. Albertson, D. Pinkel, M. V. Olson, and E. E. Eichler. Fine-scale structural variation of the human genome. *Nat Genet*, 37(7):727–32, 2005.

[148] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[149] S. Volik, S. Zhao, K. Chin, J. H. Brebner, D. R. Herndon, Q. Tao, D. Kowbel, G. Huang, A. Lapuk, W. L. Kuo, G. Magrane, P. De Jong, J. W. Gray, and C. Collins. End-sequence profiling: sequence-based analysis of aberrant genomes. *Proc Natl Acad Sci U S A*, 100(13):7696–701, 2003.

[150] V.V. Vyugin, M.S. Gelfand, and V.A. Lyubetsky. Tree conciliation: reconstruction of species phylogeny by phylogenetic gene trees. *Molecular Biology*, 36(5):650–658, 2002.

[151] P.J. Waddell, H. Kishino, and R. Ota. Phylogenetic methodology for detecting protein interactions. *Mol. Biol. Evol.*, 24:650–659, 2007.

[152] Jun Wang, Wei Wang, Ruiqiang Li, Yingrui Li, Geng Tian, Laurie Goodman, Wei Fan, Junqing Zhang, Jun Li, Juanbin Zhang, Yiran Guo, Binxiao Feng, Heng Li, Yao Lu, Xiaodong Fang, Huiqing Liang, Zhenglin Du, Dong Li, Yiqing Zhao, Yujie Hu, Zhenzhen Yang, Hancheng Zheng, Ines Hellmann, Michael Inouye, John Pool, Xin Yi, Jing Zhao, Jinjie Duan, Yan Zhou, Junjie Qin, Lijia Ma, Guoqing Li, Zhentao Yang, Guojie Zhang, Bin Yang, Chang Yu, Fang Liang, Wenjie Li, Shaochuan Li, Dawei Li, Peixiang Ni, Jue Ruan, Qibin Li, Hongmei Zhu, Dongyuan Liu, Zhike Lu, Ning Li, Guangwu Guo, Jianguo Zhang, Jia Ye, Lin Fang, Qin Hao, Quan Chen, Yu Liang, Yeyang Su, A. San, Cuo Ping, Shuang Yang, Fang Chen, Li Li, Ke Zhou, Hongkun Zheng, Yuanyuan Ren, Ling Yang, Yang Gao, Guohua Yang, Zhuo Li, Xiaoli Feng, Karsten Kristiansen, Gane Ka-Shu Wong, Rasmus Nielsen, Richard Durbin, Lars Bolund, Xiuqing Zhang, Songgang Li, Huanming Yang, and Jian Wang. The diploid genome sequence of an Asian individual. *Nature*, 456(7218):60–65, Nov 2008.

[153] René L Warren, Granger G Sutton, Steven J M Jones, and Robert A Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501, Feb 2007.

[154] David A Wheeler, Maithreyan Srinivasan, Michael Egholm, Yufeng Shen, Lei Chen, Amy McGuire, Wen He, Yi-Ju Chen, Vinod Makhijani, G. Thomas Roth, Xavier Gomes, Karrie Tartaro, Faheem Niazi, Cynthia L Turcotte, Gerard P Irzyk, James R Lupski, Craig Chinault, Xing zhi Song, Yue Liu, Ye Yuan, Lynne Nazareth, Xiang Qin, Donna M Muzny, Marcel Margulies, George M Weinstock, Richard A Gibbs, and Jonathan M Rothberg. The complete genome of an individual by massively parallel DNA sequencing. *Nature*, 452(7189):872–876, Apr 2008.

[155] David Witherspoon, Jinchuan Xing, Yuhua Zhang, W Scott Watkins, Mark Batzer, and Lynn Jorde. Mobile element scanning (me-scan) by targeted high-throughput sequencing. *BMC Genomics*, 11(1):410, 2010.

[156] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S. Kim, and D. Eisenberg. Dip, the database of interacting proteins: A research tool for studying cellular networks of protein interactions. *Nucleic Acids Res.*, 30(1):303–305, 2002.

[157] J. Xing, Y. Zhang, K. Han, A. H. Salem, S. K. Sen, C. D. Huff, Q. Zhou, E. F. Kirkness, S. Levy, M. A. Batzer, and L. B. Jorde. Mobile elements create structural variation: analysis of a complete human genome. *Genome Res.*, 19:1516–1526, Sep 2009.

[158] Kai Ye, Marcel H. Schulz, Quan Long, Rolf Apweiler, and Zemin Ning. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21):2865–2871, 1999.

[159] Deniz Yorukoglu, Faraz Hach, Lucas Swanson, Colin C. Collins, Inanc Birol, and S. Cenk Sahinalp. Dissect: detection and characterization of novel structural alterations in transcribed sequences. *Bioinformatics*, 28(12):i179–i187, June 2012.

[160] Daniel R. Zerbino and Ewan Birney. Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, May 2008.

[161] K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222, 1996.

[162] K. Zhang and T. Jiang. Some MAX-SNP-hard results concerning unordered labeled trees. *Information Processing Letters*, 49:249–254, 1994.

[163] K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42:133–139, 1992.

[164] Zhengdong Zhang, Jiang Du, Hugo Lam, Alex Abyzov, Alexander Urban, Michael Snyder, and Mark Gerstein. Identification of genomic indels and structural variations using split reads. *BMC Genomics*, 12(1):375+, 2011.