STRUCTURAL VARIATION DISCOVERY: THE EASY, THE HARD AND THE UGLY

by

Fereydoun Hormozdiari

B.Sc., Sharif University of Technology, 2004

M.Sc., Simon Fraser University, 2007

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 Science

SIMON FRASER UNIVERSITY
Fall 2011

APPROVAL

Name: Fereydoun Hormozdiari

Degree: Doctor of Philosophy

Title of thesis: Structural Variation Discovery: the easy, the hard and the ugly

Examining Committee:

Dr. Ramesh Krishnamurti
Chair

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

Dr. Evan E. Eichler, Supervisor
Genome Sciences, University of Washington

Dr. Artem Cherkasov, Supervisor
Urologic Sciences, University of British Columbia

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

Dr. Fiona Brinkman, Internal Examiner
Molecular Biology and Chemistry, Simon Fraser University

Dr. Serafim Batzoglou, External Examiner
Computer Science, Stanford University

Date Approved: **August 22nd, 2011**

Partial Copyright Licence

Abstract

Comparison of human genomes shows that along with single nucleotide polymorphisms and small indels, larger structural variants (SVs) are common. Recent studies even suggest that more base pairs are altered as a result of structural variations (including copy number variations) than as a result of single nucleotide variations or small indels. It is also been known that structural variations can cause a loss or gain of functionality and can have phenotypic effects.

Recently, with the advent of high-throughput sequencing technologies, the field of genomics has been revolutionized. The realization of high-throughput sequencing platforms now makes it feasible to detect the full spectrum of genomic variation (including SVs) among many individual genomes, including cancer patients and others suffering from diseases of genomic origin. In addition, high-throughput sequencing technologies make it possible to extend the scope of structural variation studies to a point previously unimaginable as exemplified with the 1000 Genomes Project.

In this dissertation we consider the structural variation discovery problem using high-throughput sequencing technologies. We provide combinatorial formulations for this problem under a maximum parsimony assumption, and design approximation algorithms for them. We also extend our proposed algorithms to consider conflicts between potential structural variations and resolve them. It should be noted that our algorithms are able to detect most of the well-known structural variation types including small insertions, deletions, inversions, and transpositions. Finally we extend our algorithms to allow simultaneous discovery of structural variations in multiple genomes and thus improve the final comparative results between different donors.

To My Mom and Dad And All Other Cancer Survivors

“If they Release All Political Prisoners, I will Forgive My Son’s Murderers”

— Sohrab Arabi’s Mother, ONE OF THE MARTYRS OF IRAN GREEN MOVEMENT *, June 2009*

Acknowledgments

First and foremost I offer my sincerest and utmost gratitude to my senior supervisor, Dr Cenk Sahinalp, who has supported me throughout my graduate studies with his encouragement, guidance and resourcefulness. In addition to technical skills of computer science, I have learned the requirements of being a good researcher from him. I thank him one more time for all his help.

I would like to give my regards and appreciations to Dr. Evan Eichler, his experience in genomics was a crucial factor in the success of this thesis. My visit to his lab greatly influenced the direction of my PhD research. I would also like to thank other members of my committee Dr. Inanc Birol, Dr. Artem Cherkasov and Dr. Fiona Brinkman for their help in my graduate studies.

I would like to sincerely thank Dr. Serafim Batzoglou, who kindly accepted to be my external examiner. The ideas and comments, that he shared during his visit, have provided me with additional directions for extending my research.

I would also like to thank my other collaborators Can Alkan, Iman Hajirasouliha, Phuong Dao, Raheleh Salari, Faraz Hach, Deniz Yorukoglu, and Farhad Hormozdiari. I benefited from these collaborations, and hope to continue working with them. I would also like to thank all my dear friends, Laleh Samii, Hamed Yaghoubi, Sara Ejtemaee, Shahab Tasharrofi, Mohammad Tayebi, Navid Imani, Alireza Hojjati, Azadeh Akhtari, Pashootan Vaezi, Ali Khalili, Amir Hedayaty, Hossein Jowhari, and Kamyar Khodamoradi.

Last, but not least, I thank my family for the the support they gave me in all these years. Their motivation was a huge driving force for me.

Fereydoun Hormozdiari

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the main goals of modern genomics is to discover the differences (*variation*) between genomes of two or more individuals (donors). With the advent of high-throughput sequencing technologies (also known as the next generation of sequencing technologies), the field of genomics has been revolutionized, and discovery of variants between many individuals is now achievable. We will discuss the two different strategies for the discovery of medium- and large-scale variations using high-throughput sequencing technologies. The first strategy is based on *de novo assembly* of the donor genomes, and the second strategy is based on *mapping* of the reads from donor genomes to the reference genome. For each of these two strategies many algorithms have been proposed; we will look in detail at several such algorithms and study their strengths and weaknesses.

1.1 Genome

Genome or DNA contains in most organism's their hereditary information. It is a long string of nucleic acids known as *adenine, cytosine, guanine and thymine* which are represented as *A, C, G and T* respectively. Genomes of most species are organized in a number of *chromosomes* inside the nucleus of their cells. Chromosomes can be classified into two types of autosomes and sex chromosomes. The human genome consists of 22 autosomes chromosomes (labeled chromosome 1 till 22) and a pair of sex chromosomes (labeled chromosome X and/or Y). The total length of the human genome (the combination of all the chromosomes) is around 3×10^9 nucleic acids or base pairs (bp). The first draft of the human sequence was published in 2001. It should be note that, the complete genome of a human (a combination of the genome of a few individuals) was finally published in 2003, and is being updated. This published sequence is known as the *reference genome*,

although other published genomes also existed such as HuRef [87].

1.2 Genome Variation

It is well known that differences between the genomes of each individual can cause biological difference between each individual. These differences in genomes of different individuals are known as *variants*. Genomic variations can be classified into different groups based on their size and type. The smallest yet most common type of variation is a change of a single nucleic acid known as single-nucleotide variation (SNVs). Small INDELs are larger in size compared to SNVs and constitute as insertion or deletions of few base pairs.

Structural variation (SV) are variation which effect more than 50 base pairs and can include deletions, insertions, inversions, and duplications (transpositions). Duplications of large chunks of a genome (> 1000 bp) with an identity greater than 90% are known as segmental duplications and can be categorized as a structural variation. In general any duplication or deletion of a segment of genome can be classified as a copy number variation (CNV). Segmental duplications are historical copy number variation that maybe either be fixed or polymorphic in the population.

Many of these variations between different individuals can be categorized as normal variations, which means these variations are not the underlying cause of any disease for the individual. However, it is well established that some diseases are a result of genetic variations in the effected individual. Some examples of known diseases caused by genomic variation are psoriasis [53], HIV susceptibility [43], Crohn's disease [39, 96], epilepsy [104, 50], renal disease [104], diabetes [104], autism [36], emphysema, hypertrichosis, and many more. Large insertion and deletions have also shown to have affected protein-protein interaction networks [60, 57].

A recent study by the 1000 Genome Project [1] has studied the variation in 180 different individuals as the pilot phase of a the project to discover and genotype variation in different populations. In figure 1.1 a comparison of the distribution of the size of different variation found and validated in this study is depicted. This shows the abundance of CNVs and importance of developing methods for discovery of these CNVs in different genomes. The focus of this survey is on medium and large scale genome variation and mainly in methods of discovery of such variation.

1.2.1 Non Sequencing-Based Variation Discovery

There are two main non-sequencing based methods for discovery of SVs, and in particular CNVs between a donor and the reference genome (or other donors).

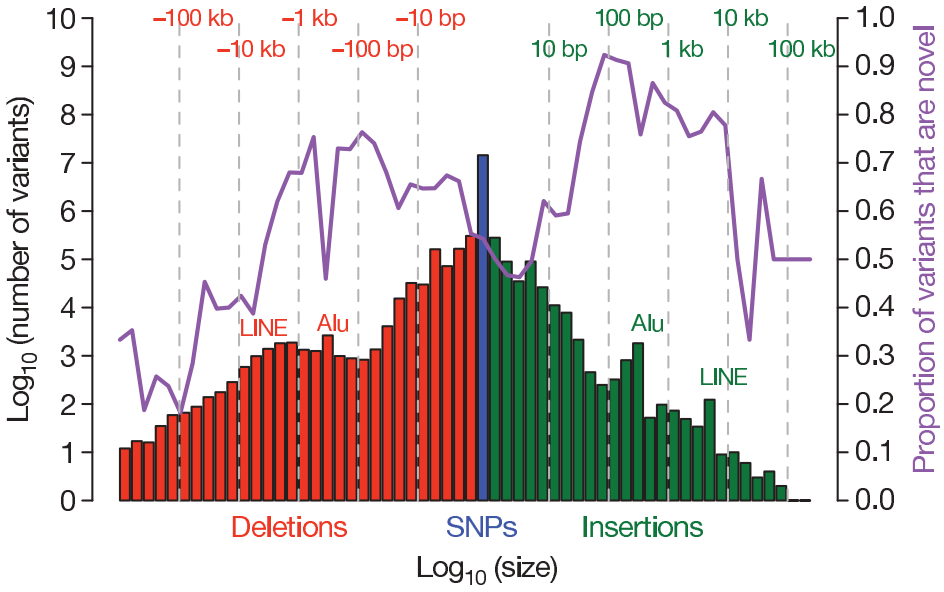


Figure 1.1: Size distribution and novelty of variation discovered by 1000 genome project [1] in comparison to the reference genome. The SNPs are shown by the colour blue, while insertions are colour green and deletions red.The purple line shows the percentage of calls which are novel in comparison to variations found in previous studies. This figure is adapted from [1].

SNP microarrays:

SNP arrays were developed to help researchers to study the single nucleotide polymorphism in human genomes. The international HapMap project used SNP arrays to detect millions of SNPs in a hundreds of individuals. The SNP arrays contain an approximate of 2 million genetic markers (in general it can vary from 250,000 to 4 million depending on the technology). The test (donor) DNA is marked using fluorescent molecules and hybridized to the array. A specialized scanners are used to detect the signal from hybridization. Recently the genotyping data from SNP array, such Affymetrix and Illumina BeadXpress is being used to detect the common and rare CNVs. Usually these methods use the allele frequency of SNPs, the distance between SNPs and the intensities of the signal to predict the SNV [3].

Array comparative genomic hybridization (aCGH):

The technology for aCGH is similar to SNP arrays, however the main objective in using aCGH is to find the ratio of difference between CNV of two different samples (affected vs. control). The oligonucleotides of the genomic region of interested are immobilized in a microarray, and the DNA

samples from the two individuals are marked with two different markers and hybridized into the array. Finally a special scanner compares the difference between signal intensity of the two colours to measure the ratio of copy number differences (please see figure 1.2). Selecting the oligonucleotides on the array is very important to reduce any hybridizations which can occur by chance [3] or minimize cross-hybridizations.

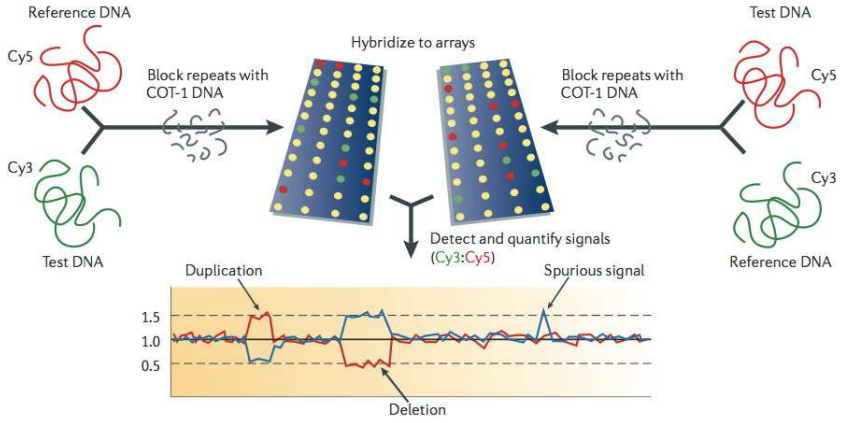


Figure 1.2: Array comparative genomic hybridization (arrayCHG) for CNV discovery between two different samples. This figure is adapted from [41]

1.3 High throughput sequencing

Till very recently the variation discovery by sequence analysis were done using low-coverage Sanger based sequencing. However in the last few years with emergence of several high-throughput sequencing platforms the field of genomics has been revolutionized. The ability to sequence genomes with high coverage and low cost is a wish come true for every researcher in field of biology and especially genomics and genetics. With advent of high throughput sequencing (HTS) technologies, detailed study of genomes of different individuals and the differences between their genomes has been made possible. Recent introduction of the high-throughput sequencing technologies has significantly changed how genomics research is conducted and has increased the demand for computational tools to optimize the utilization of the data generated by the sequencers [93]. With the arrival of these new sequencing technologies, it is becoming feasible to perform detailed and comprehensive genome variation and rearrangement studies. High-throughput, low-cost sequencing technologies such as pyrosequencing (454 Life science [Roche]), sequencing-by-synthesis (Illumina), and

sequencing-by-ligation (ABI SOLiD) methods produce much shorter reads than the traditional capillary sequencing, but they also increase the redundancy by 10-to 100 fold or more [135, 93]. Most of the high-throughput sequencing technologies have the ability to produce mate-pairs or paired-ends reads. For production of the paired-end reads the genome should be fragmented into small pieces and the two end of the fragments each sequenced by the high-throughput sequencers. The length of the fragments produced usually obeys a tight normal distribution which is crucial for any variation discovery method.

The (paired-end) reads provided by high throughput sequencers such as Illumina, 454 or SOLiD machines can be used to discover variations by two main frameworks. The first framework is based on using a de novo assembly algorithm (such as Euler [114, 115] or ABySS [136]) to create large contigs, and then align the contigs to the reference genome to discover different variations between the donor genome and the reference genome. The second framework is based on using the reads provided by the sequencer directly in conjunction to the reference genome to predict different variation (the first frameworks for this strategy were published in [144, 147, 9]).

Each of the strategies and methods developed for structural variation discover is able to discover variation of different size. The HTS methods have the highest resolution in finding the smallest SV, while array platform (i.e. SNP microarray and aCGH) are not able to detect the smaller CNV which the HTS methods are able to capture. Fosmid mappings and BAC arrays are useful to find the very large CNV. In the figure 1.3 a detailed comparison of size of different CNV detection methods is provided.

1.4 De novo assembly and comparison

One of the main approaches used to find variation between different genomes is to directly compare their (assembled) genomes with each other. This approach has two steps, first by utilizing a de novo assembler (such as Euler [114]) larger contigs are created by concatenating [1] the reads produced by the sequencer *(assembly phase)*. The second step is to align the contigs produced in the first step to the reference genome to predict different variations. Both these steps have been studied extensively and major progress has been made on them.

[1] the assembly algorithms are quite complex and we will go over them in next section

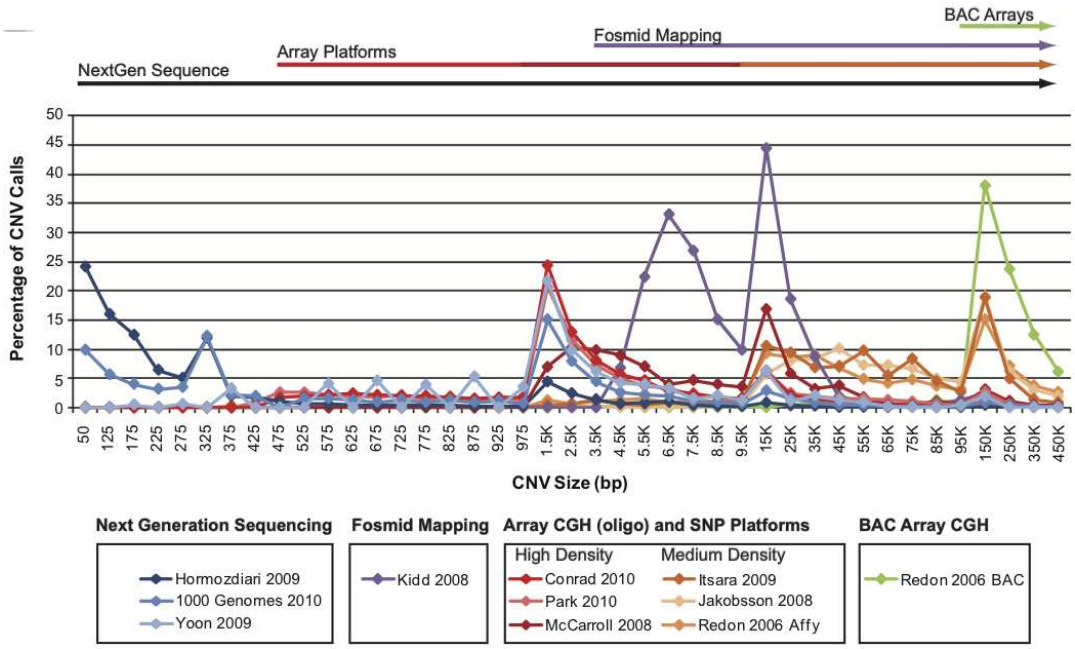


Figure 1.3: Different technologies for CNV discovery are able to discover structural variations with different size. This figure shows the ability of each platform in discovery of events of different size. This figure is adapted from [3]

1.4.1 De novo assembly

One of the first formulations for de novo assembly was *shortest superstring problem*. Shortest superstring problem is defined as follow.

shortest superstring : Given a finite alphabet σ and a set of n strings $S = \{s_1, s_2, \cdots, s_n\}$, find the shortest string s that contains each s_i as a substring. In the context of the genome assembly $\sigma = \{A, C, G, T\}$ and the set of strings S is the set of reads produced by the sequencer.

The shortest superstring problem is known to be NP-hard, even for the special case that $\sigma = \{0, 1\}$. The simplest algorithm which one can come up with for this problem is a naive greedy method. The algorithm is as follows: iteratively merge two strings with maximum overlap and replace the merged string with those two strings until you are left with only one long string. Although the approximation factor for this algorithm is not known, however it is conjectured to be 2 [145].

There are few approximate algorithms proposed for this problem. One of the less effective approximation algorithms for this problem provides an approximation factor of $2H_n$ which is asymptotically equal to $2 \log n$, this method is based on greedy method proposed for set-cover.

There are methods proposed which are able to achieve constant factor approximation for shortest superstring problem. Here we explain the algorithm which has approximation factor of 4 (this algorithm can be improved to yield an approximation factor of 3). First build a prefix graph $PG(V, E)$ of the strings in set S. A prefix graph is a weighted directed graph where each node represents a string from set S and weight of each edge connecting s_i to s_j is $|prefix(s_i, s_j)|$ (here $|prefix(s_i, s_j)|$ the length of the prefix of s_i when the maximum possible overlap with a suffix of s_j is removed). The minimum traveling sales person (TSP) on the prefix graphs is a lower bound on the shortest superstring, however TSP problem can not be approximated (note if TSP had an approximate algorithm, it can be used for deciding the hamiltonian cycle problem which is NP-hard). Instead of TSP on the prefix graph, consider the minimum cycle cover on the prefix graph which is solvable efficiently. For a given prefix graph $PG(V, E)$, build a bipartite graph $G'(V', U', E')$ where each node in V is represented by a node in each sides of the bipartite graph G'. Assume set $U' = \{u_1, \cdots, u_n\}$ and $V' = \{v_1, \cdots .v_n\}$ are the vertices of two sides of bipartite graph G', and the weight of edge connecting u_i and v_i is equal to the $|prefix(s_i, s_j)|$. Now a minimum perfect matching on G' is equal to a minimum weighted cycle cover in $PG(V, E)$. For each cycle $c = (i_1, i_2, \cdots, i_\ell, i_1)$ selected as a member of minimum weighted cycle cover in the prefix graph $PG(V, E)$, select an arbitrary node such as i_1 and produce the string $\delta(c) = prefix(s_{i_1}, s_{i_2}) \circ \cdots \circ prefix(s_{i_\ell}, s_{i_1}) \circ s_{i_1}$. The operation $\circ$ is a concatenation operation on two strings. Concatenation of the $\delta(c)$ for all of the minimum cycle covers of the prefix graph gives an approximation factor of 4 for shortest superstring problem [20, 145].

Although there exists approximation algorithms with constant approximation factor for shortest superstring problem, however in practice these methods were not accurate enough and thus are not used for de novo assembly. In addition the relaxation of de novo assembly problem to shortest superstring problem has the drawback of collapsing all the repeats in the donor genome which is not desirable. There are two main modern de novo assembly approaches which have been relatively proven effective in practice, one is known as *overlap-layout-consensus* approach, and the other is *de bruijn graph* approach.

overlap-layout-consensus

This framework has three stages as suggested by its name. In the first stage, every pair of read is compared with each other and an overlap metric between them is calculated. This stage is known as the *overlap step*. In the second stage (i.e. *layout* stage) the goal is to order overlapping reads, such that they form larger sequences called contigs. Finally in the third stage (i.e. *consensus* stage) the

computed layout in the previous stage is realized by constructing the assembled sequence.

Most methods which adopt this approach are using directed graphs to model the problem. In the first step an overlap graph G is build from the set of reads provided. In G each read is represented by a node and there is a directed edge between two nodes if there is a significant overlap between them (the suffix-prefix relationship between two reads indicates the direction of the edge). In the layout step the goal is to find a path which goes over all the nodes *exactly* once. Thus the layout step is equivalent to finding an hamiltonian path in graph G, which is NP-hard problem. Several heuristics are used in the layout step to simplify the graph and find a path which covers most of the nodes. Finally in the consensus step the assembled sequence from traversing the path found in previous step is produced. This approach can be extended to use paired-end read data for assembly [16].

De bruijn graph - Eulerian path model

A closely related problem to shortest superstring problem is known as *sequencing by hybridization problem*. First, define the l-spectrum of a string S to be the set of l-mers in S. In sequencing by hybridization problem, we are given a set of l-mers $\mathcal{S}$ and the goal is to find a shortest string S whose l-spectrum is exactly $\mathcal{S}$. For solving this problem "de bruijn" graphs are used. The set of vertices of the de bruijn graph is $\mathcal{S}$. There is a directed edge between two vertices u and v in the graph if the last $l-1$ characters of u is equal to the first $l-1$ characters of v. A solution to the *sequencing by hybridization problem* consists in a shortest Eulerian path in the de bruijn graph.

The output of sequencing is reads, not the l-spectrum. Therefore, in de bruijn graph based framework the reads are shredded into their constituent l-mers, and the de bruijn graph G is built on the union of the l-spectrums of the reads. Reads are paths in the graph, and a solution to the assembly problem is an Eulerian path that has all reads as subpaths. Finding such an Eulerian path is called *Eulerian super path problem* [115].

Let $P = \{p_1, \cdots, p_n\}$ be the set of paths in G specified by the reads. A transformation from graph G and a set of paths P to graph G_1 and a set of paths P_1 is called equivalent if there exists a one-to-one correspondence between Eulerian superpaths in (G, P) and those in (G_1, P_1). The goal is to find (G_k, P_k) after k transformations from (G, P) such that every path in P_k is an edge in G_k (a set of equivalent transformations is given in [115]). The Eulerian superpath problem is NP-Hard in general, therefore, equivalent transformations are not guaranteed to solve the problem [103].

Many extensions of de bruijn graph for de novo assembly problem have been proposed, which have improved the results significantly. For instance this model is extended to work with paired-end data [115, 142], short reads produced by high-throughput sequencers [26, 159], and making the

assembler parallel on clusters for large genomes [136].

ABySS: The standard implementations of de bruijn graph assemblers are not scalable for large genomes such as human (an approximate length of 3×10^9 bp), thus there is a need for parallelized implementation of this algorithm. ABySS [136] is an assembler which is capable to scale up the de bruijn graph assembler to large genomes using a cluster of computers. In this method the adjacent nodes need not be in the same computer, which allows the nodes to be in different computers and thus making usage of a distributed systems possible for genome assembly. The main idea behind this method is to distribute the nodes of the de bruijn graph into different computers such that they can be accessed in $O(1)$ time; thus the id of the computer which each node lies in should be calculable in constant time. For each l-mer (node) we need two pieces of information, one is in which computer in the cluster the node resides in, and second the adjacency information. For each node (l-mer) a simple hash function is used to find the id of the computer which the node is stored in, and the adjacency information between two nodes is stored in compact representation. Considering a node there are at most 8 possible edges (4 incoming and 4 outgoing - assuming our alphabet set is $\Sigma = \{A, C, G, T\}$), existence of each of these adjacency edges can be stored in one bit (and thus a total of 8 bits per node is necessary to keep the adjacency information of each node).

String Graphs: An alternative formulation for de bruijn graphs was proposed by Myers [110] which is known as string graphs. String graphs are similar to overlap graphs. For each read a node is considered and overlaps between nodes is corresponded by edges (the prefix of one read is the suffix of the other). The string graph is build from the overlap graph by process of transitively inferable edge reduction. For each three node x, y and z, if there is an edge from y and z to x, and there is an edge from z to y the the edge from z to x is considered redundant and remove. In [110] using a statistical framework edges are classified as optional, required and exact. The goal is to find the shortest path which goes through all the exact edges only one time, over all the optional edges any number of times and over required edges one or more times. This problem is also NP-hard as shown in [103].

1.4.2 Edit distance and block edit distance

Finding the *minimum* differences between two strings S and R of alphabet $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_\ell\}$ is an extremely important problem. There are different variations of this problem which can be studied based on the edit operations permitted. The most basic edit operation is substitution, in this

operation a character from alphabet σ in sequence S is replaced with another character. Hamming distance is defined as minimum number of substitutions which can transfer string S to R. The hamming distance between two strings can be found easily in linear time. The most dominant edit operations in addition to single character substitutions is single character insertions and deletions. Standard edit distance is defined as minimum number of such operations which transfers string S to string R (this distance is also known as levenshtein metric). The algorithms for finding edit distance are more complicated than hamming distance algorithms, and usually are based on dynamic programming techniques. The standard edit distance between two strings S and R can be computed in $O(nm)$, where n is length of string S and m is length of string R, using a dynamic programming. A similar dynamic programming is used in Needleman-Wunsch algorithm for global alignment, and Smith-Waterman algorithm for local alignment. It is not hard to see that if the edit distance between sequence R and S is known to be sublinear (lets denote it by $ED(R, S)$) and less than a threshold t, the dynamic programming method can be modified to yield a running time of $O(n \cdot t)$. A popular extension of edit distance is edit distance with affine gap penalty. In this model the start of the gap has different penalty than the extension of the gap. This model of edit distance is well motivated from biology.

The above distances are only able to characterize variation which effect single characters. As mentioned before, there are variation of interest which effect substrings, such as deletion of a substring, insertions of a substring or inversion of a substring (these operations are known as *block edit operations* in general, and structural variation in genomes) [109, 22]. Unfortunately finding the minimum number of edit operations which transfer sequence S to R is NP-complete when block operations are permitted [124, 32] . For instance problem of finding minimum number of block transpositions (move) which convert sequence R to sequence S is NP-complete (even when R and S are permutations). This problem is known as edit distance with moves ($EDM(R, S)$), and in special case of permutations it is denoted simply as permutation edit distance ($PED(R, S)$) [2]. The problem of minimum number of move operation on permutations can be approximated with a factor of 1.5 [7]. The problem of finding minimum number of operations, when we are permitted to have block move, inversion, insertion and deletions is also NP-complete. This problem is denoted as block edit distance $BED(R, S)$ between two string R and S [124].

One of the first problems considered by researchers was minimum reversal problem on permutations (this problem is also known as reversal distance problem). The motivation behind considering

[2]These notations are borrowed from [124]

permutations comes from two assumptions, first assumption was that each gene is completely different from another gene and thus each gene can be represented as a distinct character in our permutation; second assumption was that no edit operation which changes/disrupts a gene happens in nature (both these assumptions are currently known to be false). The problem is defined as follows: given two permutations from the same set of alphabet, find a series of reversals on one of the permutations which transfers it to the other permutation. This problem is equivalent to another problem known as *sorting by reversal*. In sorting by reversal we are given a single permutations and would like to find minimum number of reversals to transfer the input permutation into the identity permutation. Sorting by permutation is known to be NP-hard problem and approximation algorithms are known for it. Here we would explain an algorithm for this problem which gives an approximation factor of 4 [69]. Before giving the details of the algorithm we need to define few notations which will be used in the explanation of the algorithm. Each pair of neighbour elements in the permutation which are not consecutive numbers is called a *breakpoint*. The algorithm in each iteration picks in a greedy fashion a reversal which minimizes the number of breakpoints [3] (note that only a sorted permutation has no breakpoint). It can be shown that if there exists a decreasing stretch of elements in the permutation, then there exists a reversal which reduces number of breakpoints (please see [69] for the details). In addition if there does not exist a decreasing stretch of elements then we can reverse one of the increasing stretches (this guarantees that we can find a reversal which decreases the total number of breakpoints). The above algorithm is shown to have an approximation factor of 4. The best approximate algorithm for this problem is introduced in [19], with approximation factor of 1.375.

The more general edit distance with block operations has also been studied in context of approximation algorithms. A seminal work by Muthukrishnan and Sahinalp [109] has provided an algorithm to approximate the $BED(S,R)$ with an approximation factor of $\log n \cdot \log^* n$ ($\log^* n$ is number of recursive $\log$ functions which need to be applied to n so it becomes less than 1 [4]). The proposed algorithm of Muthukrishnan and Sahinalp [109], embeds the strings using a *local consistency parsing* into hamming distance with distortion factor of $\log n \cdot \log^* n$.

[3] In another words in each iteration the reversal which reduces maximum number of breakpoints in the permutation is considered

[4] $\log^* n = min\{i \geq 0 : \log^i n \leq 1\}$

1.5 Mapping based strategy

Another approach used to discover variation between individuals is by using the reference genome as intermediate guide. This approach usually has two step:

(I) in the first step reads from the donor genome are mapped (aligned) back to the reference genome; loci of the reference genome with sequence similarity more than a user defined threshold in comparison to each read are considered as potential regions that the read is sequenced from.

(II) in the second step abnormalities in the mapping is used to make predictions about variation on the donor genome.

This framework is known as mapping based. Most of the mapping based methods were designed primarily to be used with capillary sequencing. Thus, these methods should be modified (extended) before we can apply them to genomes sequenced using high throughput sequencing technologies.

Here we study main frameworks and methods used for finding different types of structural variation in mapping based strategies. There are three main frameworks used for SV detection using high-throughput technologies:

1. Read-Pair Analysis : mainly used for discovery of different types of variation, including deletion, insertion, inversion, translocation and mobile element insertions.

2. Read-Depth Analysis : mainly used for copy number variation discovery.

3. Split-Read Analysis : mainly used for discovery of deletion and small insertions, and is in particular useful in finding the exact breakpoint of the variation.

Each type of structural variation is best found using some of the strategies explored by the researchers. In figure 1.4 a summary of proposed approaches for different types of structural variation discovery is provided. At the core of all these frameworks is mapping of paired-end reads to the reference genome. In table 2, a high level comparison of different available methods for SV discovery using high-throughput sequencing technologies is provided. Note that the main limitation of these methods is dependence on the reference genome. Before giving the details of some of these mapping based algorithms we would like to look at mapping algorithms, which are the first step of most of these methods.

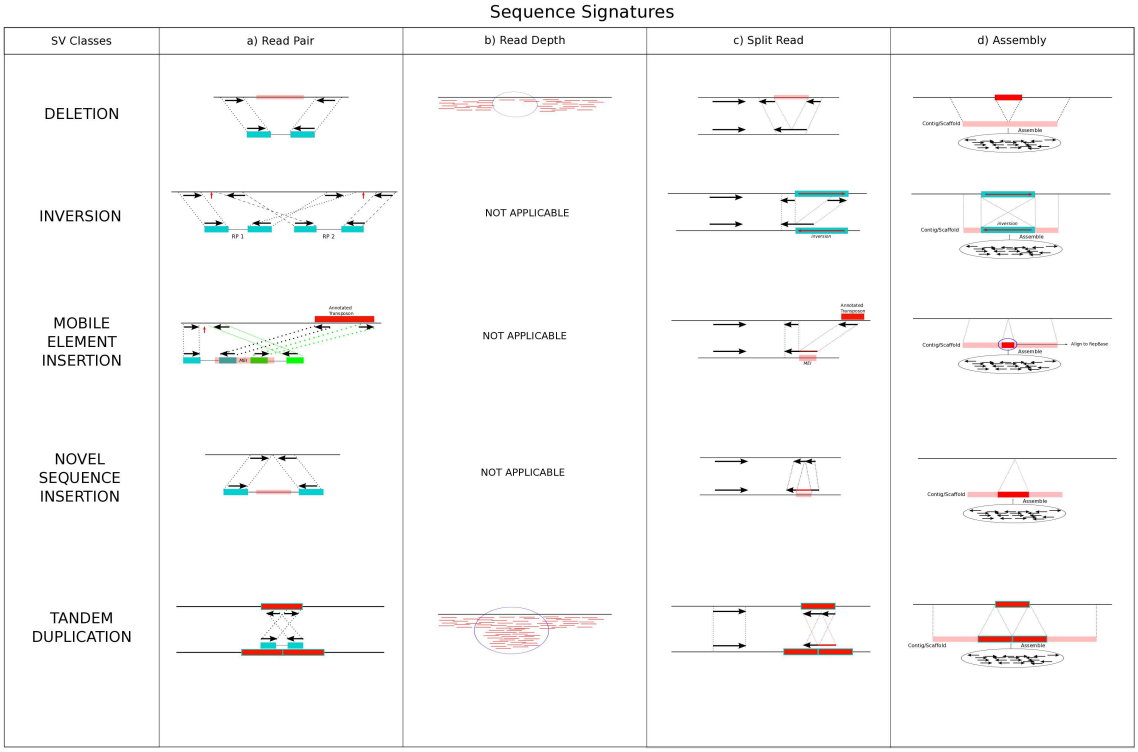


Figure 1.4: Four major sequenced-based approaches to detect structural variation are shown in this figure. The read-pair, split-read and de novo assembly methods can find different SV types, while read-depth can only discover loss or gain of a genomic region. Figure adapted from [3].

Algorithm	Method	Capability	Platform	Strategy	SV Classes				
					Del.	Inv.	Novel Ins.	Mobile Elem.	Dup.
VariationHunter [55]	RP	Discovery	Illumina	Maximal Clustering and maximum Parsimony	50bp-500Kbp	Yes	Yes	Yes	No
PEMer [78]	RP	Discovery	454	Outlier clustering	300bp-1Mbp	Yes	Yes	No	No
BreakDancer [27]	RP	Discovery	Illumina	Outlier clustering and distribution analysis	10bp-1Mbp	Yes	Yes	No	No
MoDIL [85]	RP	Discovery	Illumina	Outlier fitting distribution analysis	6bp-70 Kbp	Yes	Yes	No	No
CNVer [102]	RP/RD	Discovery Genotyping	Illumina	Outlier fitting and probabilistic graph analysis	1 Kbp - 550Kbp	No	No	No	Yes
Spanner [139]	RP/RD	Discovery Genotyping	Illumina 454	Outlier clustering	10 bp - 200 Kbp	?	yes	yes	tandem Dup.
SLOPE	RP	Discovery	Illumina		10 bp - 1 Mbp	no	yes	no	no
Hydra [118]	RP	Discovery	Illumina	maximal Clustering	50bp-1Mbp	Yes	Yes	Yes	No
SVDetect [158]	RP	Discovery	Illumina SOLiD	clustering and Sliding Window	50bp-1Mbp	Yes	Yes	No	Yes
GASV [137]	RP	Discovery	Illumina SOLiD	Maximal Clustering	50bp-1Mbp	Yes	No	No	No
Corona Light [98]	RP	Discovery	SOLiD	Outlier clustering and significance testing	5 bp - 85 Kbp	No	Yes	No	No
mrCaNaVar [4]	RD	Discovery Genotyping	Illumina 454	Distribution-based clustering	10 kb-10Mbp	No	No	No	Yes
EWT [157]	RD	Discovery	Illumina	Significance-testing	200bp-250Kbp	No	No	No	Yes
CNVnator [2]	RD	Discovery Genotyping	Illumina	Mean-shift process	100bp - 1.5 Mbp	No	No	No	Yes
Pindel [156]	SR	Discovery	Illumina	Pattern growth	1bp-50Kbp	No	1bp-20bp	No	No
YALE-SR	SR	Discovery	454	Gapped alignment	51bp-750Kbp	Yes	51bp-336	No	No
NovelSeq [47]	AS	Discovery	Illumina	local assembly and maximum matching	No	No	$>$ 200bp	No	No
Cortex	AS	Discovery	Illumina 454	de Bruijn graph	30 bp - 40 Kbp	No	$>$ 50bp	yes	No
SOAPdenovo [91]	AS	Discovery	Illumina	de Bruijn graph	50 bp - 4 Kbp	No	$>$ 51bp	yes	No
BreakSeq [81]	Breakpoint library	Genotyping breakpoints	any	Read mapping to	yes	yes	yes	yes	yes

Table 1.1: In this table a summarization of different methods used for structural variation discovery based on the read pair, read depth and split read analysis is provided.

1.5.1 Mapping Algorithms

In this section we will provide a short introduction on two sets of mapping strategies which are currently popular for mapping reads produced by high-throughput sequencing technologies to the reference genome.

The mapping problem can be formalized as follow: Given a text T (i.e. the reference genome) and a search pattern P (i.e. reads from the sequencer), the goal is to find matches with "distance" less than a threshold k. The distance metric usually used are edit distance or hamming distance.

It is obvious that a simple global alignment of the pattern to the text can find all the locations with the desired constraint. However, the high running time of checking all the locations encourages us to develop new algorithms to prune out cases which are not promising in constant time. One of the most popular strategies used for mapping the reads to the reference genome is based on seed and extend approach [5], the other recently popular strategy uses the Ferragina-Manzini index (FMI) [40] to map the reads [6].

In the seed and extend strategy a hash table on the text is build and is used to prune cases which can not satisfy the mapping requirement in constant time (seed step). In the next step, among the loci in text (i.e. the reference genome) which are considered as potential candidates, Needleman-Wunsch algorithm is ran to choose the loci with distance less than k (extend step).

In many of the seed and extend algorithms a simple fact is used to perform the pruning while not sacrificing the accuracy. A pattern of length m can be aligned to text with edit distance less than or equal to k if there exists a substring of length $\lfloor \frac{m}{k+1} \rfloor$ from the pattern which can match the text exactly. For simplicity of notations lets represent the value $\lfloor \frac{m}{k+1} \rfloor$ as ℓ. Seed and extend algorithms work by first anchoring ℓ-mers (seeds) from the read to the reference genome, by interrogating an index (in the form of a hash table for all ℓ-mers and their respected loci from reference genome) and then extending the anchored loci by allowing at most a user-specified number of mismatches or indels (i.e maximum edit distance k).

MrFAST, MrsFAST and DrFast algorithms

MrFAST (micro-read fast alignment search tool) [4] implements a collision-free hash table to create indices of the reference genome. Each ℓ-mer (ℓ usually is in the range of 10 to 12 bp) of the reference

[5] Algorithms such as MAQ, Mosaik, Slider, mrFast, mrsFast use this approach to map the paired-end reads to the reference genome

[6] Algorithms such as BWA and Bowtie use this approach

genome is included in an array using a hash function which can access a particular ℓ-mer in constant time. Each loci in the reference genome is stored in the hash table. The main advantage of MrFAST over other seed and extend methods is its fast way of verifying the loci returned after anchoring ℓ-mers of the pattern against the text. Normally we can use a Needleman-Wunsch algorithm to compare the loci returned after anchoring the ℓ-mer of the pattern with running time of $O(m^2)$. However we are only interested in matches with total edit distance less than k, thus we can use a well known strategy (banded dynamic programming) to only complete a small portion of the dynamic table and decrease the running time to only $O(mk)$. Another trick used in MrFAST to improve the running time was the implementation of the dynamic programming using SSE2 instruction set. Basically, through the use of the SSE2 instruction set we can calculate values of multiple data point with only a single instruction. More specifically, in the dynamic programming table we can calculate values of a diagonal of the table with one operation (instead of $2k-1$ different operations for each cell in the diagonal of the dynamic programming matrix).

MrsFast [46] is similar to MrFast with two main differences. The first difference is that in MrFast only the ℓ-mers of the text are hashed in an array, while in MrsFast the ℓ-mers of the text and the reads are hashed. The second difference is that MrsFast is a *cache oblivious* algorithm. In another words it tries to minimize the total number of cache misses and thus improve the running time of the algorithm. To our knowledge MrsFast is the first method which introduces optimizing cache performance for mapping, which improves the results significantly [46]. DrFast [54] is an extension of above algorithms for color-space mapping (for ABI-Solid data).

Bowtie and BWA algorithms

Another recently popular approach for mapping paired-end reads back to the reference genome is based on Burrows-Wheeler transform (BWT) which uses the Ferragina-Manzini index (FMI). This method is been used in Bowtie and BWA software.

The BWT is a reversible reordering of the text, which is used in data compression. Given a text T the Burrows-Wheeler forward transformation of the text is denoted as $BTW(T)$. The algorithm to produce the transformation works as follow: (1) Append to the end of the text T a special character \$ (which is considered lexicographically smaller than any other character in Σ); (2) Build a matrix M whose rows are cyclic shifts of $T\$$ and lexicographically sort the rows; (3) the last column of the matrix M is the $BWT(T)$.

In [40] an algorithm for searching a pattern against the BWT transformation is provided. In BWA and Bowtie heuristics are provided to use the $BWT(T)$ to search a given pattern P against

text T with a given threshold of edit distance. These heuristics are exponential on the edit distance allowed (i.e. k), and thus will not be very efficient in practice for longer reads (as shown in [46]).

1.5.2 Read-Pair Analysis

The general ideas for structural variation discovery between the donor genome sequenced by capillary sequencers and the reference genome using read-pair analysis was introduced in [147, 144]. The read-pair analysis is based on aligning the sequenced paired-end reads to the reference genome and observing significant differences between the distance of *end-pairs*[7] mapped to reference genome in comparison to the expected distance - which indicates a *deletion* or an *insertion* event. Furthermore, one can also deduce inversion events: if one of the two ends of a pair has a different orientation, this is likely a result of an *inversion* (see [144]). In case, the two ends are *everted*, i.e. both ends of a paired-end have reverse orientation (with respect to each other), but their order is preserved in the reference genome, we are likely to have a *tandem repeat* [55]. Finally, paired-end reads mapping to two different chromosomes are likely to be a result of a *trans-chromosomal* event (see Figure 3.1).

At core of the above general strategy is the computation of the expected distance between end-pairs in the donor genome, which is referred to as insert size ($InsSize$). Previous works [144, 79, 84] assume that for all paired-end reads, $InsSize$ is in range of $[minLen, maxLen]$ which can be calculated as described in [144]. The $minLen$ and $maxLen$ are assumed to be $mv \pm \{3, 4\} \times std$, when mv is the mean value and std is the standard deviation of the distance between two ends of the paired-end reads mapped to the reference genome (after discarding the outliers).

An alignment of a paired-end read to the reference genome is called *concordant* [144], if the distance between end-pairs is in the range of $[minLen, maxLen]$, and both the orientation and the chromosome that the paired-end read is aligned to are "correct". A paired-end read which has no concordant alignment in the reference genome was defined in [144] and later in [79, 84], and is called a *discordant* paired-end read (which indicates a possibility of a structural variation). One of the first methods to use high-throughput sequencing technologies to find large scale variation was proposed in [79]. In this paper fragments of median 3kbp were sequenced by 454 technology, and mapped to the reference genome. Using a strategy similar to [144] structural variation of type deletion, inversion and insertions were predicted.

[7] End-pairs refer to the two ends of a paired-end read.

BreakDancer [27]

BreakDancer is claimed to be able to detect variation as small as 10bp and as large as 1Mbp using read-pair analysis. The method itself consists of two parts, BreakDancerMax and BreakDancerMini. BreakDancerMax is able to predict medium and large size variation, including insertion, deletion, inversion and translocation, while BreakDancerMini predicts small deletions of size ranging from 10bp to 100bp. The BreakDancerMax uses best mappings returned for paired-end reads with high quality for structural variation predictions. Each paired-end read is classified to six different classes based on the mapping. The classes include normal, deletion, insertion, inversion, intrachromosomal translocation and interchromosomal translocation. The normal class is equivalent to the concordant paired-end reads as defined in [144]. These classifications are done based on the distance between the two ends of the paired-end reads mapped, the orientation of the mapping, and expected insert size of the paired-end reads. The paired-end reads which are not classified as normal (i.e. concordant) are called anomalous paired-end reads (i.e. discordant). The algorithm searches for regions of genome where two or more anomalous paired-end reads are mapped. Each potential variant is given a confidence score based on number of paired-end reads supporting it, the size of the region and coverage of the sequenced genome.

The BreakDancerMini is responsible to find deletions which are smaller than 100bp and would not be found using BreakDancerMax. In BreakDancerMini paired-end reads which were classified as normal are considered further to detect deletions which are smaller than the range of deletions which can be found using anomalous paired-end reads. The BreakDancerMini algorithm uses a sliding window approach to classify every window of length equal to mean insert size to anomalous region or normal region. It compares the distance between two end of the paired-end reads mapped inside the window to the paired-end reads mapped outside of the window. For a region which is classified as anomalous region a confidence score is assigned similar to BreakDanceMax as indicator of significance of the sliding window test.

Multiple mapping for read-pair analysis [84]

One of the first algorithms that utilizes multiple mappings of paired-end reads to detect structural variation was introduced in [84]. In [84] a probabilistic framework for finding structural variations (insertion, deletion, inversion and translocation) was proposed. A set of paired-end read mappings, which explain the same structural variation, is called a cluster and denoted by C_k; the probability of a paired-end read mapping X_i explaining the same variant as the cluster C_k is denoted as

$P(X_i|C_k)$ and $P(C_k)$ denotes the probability of cluster C_k being a real cluster and representing a true SV. The framework assigns probability to each pair of paired-end reads (such as X_i and X_j), if the two reads can be part of the same cluster C_k (denotes by $P(X_i, X_j|C_k)$). To calculate the probabilities, they use independence between paired-end read mappings and thus conclude $P(X_i, X_j|C_k) = P(X_i|C_k)P(X_j|C_k)$. The first step of the algorithm is the clustering of the paired-end reads to identify all of the potential structural variants. The clustering method used by [84] is a simple hierarchical clustering which is based on greedy strategy. First step every paired-end read mapping is in a clusters for itself, and in each step two clusters with the smallest "distance" separating them are merged. This procedure continues till no two cluster are closer than a user defined threshold. The distance between two clusters C_u and C_v is calculated as $D(C_u, C_v) = \frac{1}{|C_u||C_v|} \sum_{X_i \in C_u, X_j \in C_v} \log P(X_i, X_j|C_m)$ where C_m is the cluster consisting of X_i and X_j. Note that using this method of clustering every paired-end read mapping is in exactly one cluster. After clustering of the paired-end read mappings the algorithm chooses a unique mapping location for each paired-end read. [84] defines a probabilistically motivated optimization function $J(w)$ to assign each of the paired-end reads to only one location. The optimization function $J(w)$ is linear combination of three different scores, which represent the sequence similarity of each mapping, probability of each cluster being valid and cardinality of each cluster (for more details please see [84]). For finding the optimal set of assignment to maximize the optimization function $J(w)$ a hill climbing approach is used. This hill climbing approach in each step picks a paired-end read which can be moved from its current cluster to another cluster such that it increases the optimization function.

Gene fusions [12]

One of the very important functional consequences of structural variation are gene fusions. Fusion genes are hybrid of two genes which are caused by an SV (i.e. inversion, deletion or translocation) which effects two normal genes.

The method proposed in [12] to discover gene fusions uses a strategy similar to the one proposed in [144]. Mappings of discordant paired-end reads, where the two ends (more accurately the two breakpoints) overlap two known genes, are considered as potential indicator of a gene fusion. [12] method like many other paired-end read methods has an implicit assumption that each paired-end read has only one mapping with edit distance less than a user defined threshold.

A simple set of rules is used to calculate the potential breakpoint(s) suggested by each discordant paired-end read mapping. Assume for each discordant paired-end read c the two ends are mapped to

locations x_c and y_c in the reference genome (w.l.o.g $x_c < y_c$). Given a discordant paired-end read c the possible breakpoints a and b of the potential SV supported by c can be calculated as provided below. If the paired-end read c is supporting an inversion and the two ends are mapped to forward orientations then the potential breakpoint a and b can be calculated as follows:

$$minLen \leq (a - x_c) + (b - y_c) \leq maxLen$$

If the paired-end read c is supporting an inversion and the two ends are mapped to reverse orientations then the potential breakpoint a and b can be calculated as follows:

$$minLen \leq (x_c - a) + (y_c - b) \leq maxLen$$

If the paired-end read c is supporting a deletion then the potential breakpoint a and b can be calculated as follows:

$$minLen \leq (a - x_c) + (y_c - b) \leq maxLen$$

The breakpoints (a, b) for a clone c in a two dimensional space will define a trapezoid (if $minLen = 0$ it will be a triangle). A set of paired-end reads mappings is defined to be a *cluster* if the intersection of the trapezoids of the possible breakpoints of paired-end reads is not empty. They calculate the probability of a gene fusion which spans two genomic regions (e.g. U and V) as explained below.

Lets define event $C_{(a,b)}$ as the event of a discordant paired-end read C supporting an inversion at breakpoint (a, b) (with mapping coordinate of (x_c, y_c)). Assume w.l.o.g. $a \geq x_c$ and $b \geq y_c$. The length of the clone C can easily be calculated as $l_C(a, b) = a + b - x_C - y_C$. Thus the probability of breakpoint (a, b) can be expressed in terms of distribution of length of paired-end reads.

Now consider two genes which span two genomic regions of U and V, probability of a gene fusion between these two genes given a clone C is equal to the probability of the breakpoints supported by clone C (e.g. (a, b)) are in region $U \times V$. More formally as defined in [12], the event $C_{(a,b)}$ implies the event of size of clone C being equal to $l_C(a, b)$. Lets denote for simplicity $N_C[s]$ as number of discordant breakpoints (α, β) where $\alpha \geq x_C$ and $\beta \geq y_C$ and $\alpha + \beta = s$. The equation below follows from the assumption that all the breakpoints are equally possible

$$P(C_{(a,b)}) = P(C_{(a,b)} \cap (L_C = l_C(a, b))) = \frac{1}{N_C[a + b]} P(L_C = l_C(a, b))$$

Probability of gene fusion between two genes which span genomic regions U and V which are spanned by clone C, given clone C can be calculated by taking the summation over all possible breakpoints of fusion in regions U and V given clone C as :

$$P(\cup_{(a,b)\in U\times V} C_{(a,b)}) = \sum_{(a,b)\in U\times V} \frac{P(L_C = l_C(a,b))}{N_C[a+b]}$$

Finally the probability of gene fusion from single clone C is extended to a set of clones $CC = \{C^{(1)}, C^{(2)}, \cdots\}$ which overlap the same breakpoint (a,b). Define the event $\hbar$ as the event that all the clones in set CC overlap the same breakpoint. Lets define the event $\hbar = \cup_{(a,b)} \hbar_{(a,b)}$ where $\hbar_{(a,b)} = \cap_j C^{(j)}_{(a,b)}$ as the event where all the clones in set CC overlap the breakpoint (a,b). In [12] it was shown that

$$P(\hbar_{(a,b)}|\hbar) = \frac{\prod_j P(C^{(j)}_{(a,b)})}{\sum_{(a,b)} \prod_j P^{(j)}_{(a,b)}}$$

and probability of gene U and V being fused is

$$P(\cup_{(a,b)\in U\times V} \hbar_{(a,b)}|\hbar) = \frac{\sum_{(a,b)\in U\times V} \prod_j P(C^{(j)}_{(a,b)})}{\sum_{(a,b)} \prod_j P(C^{(j)}_{(a,b)})}$$

SPANNER

Another algorithm which uses the read-pair analysis to find structural variation (deletion and duplications) is known as SPANNER [139]. In first step of this algorithm paired-end reads which are duplicated in process of sequencing are removed (i.e. if two paired-end read have the same mapping positions for both ends only one is considered), also any paired-end read with average phred quality value less than 30 is discarded. In second step paired-end reads which support the same variation are clustered together using a nearest neighbour clustering algorithm such that reads that are in the same cluster support the same breakpoints. Clustering is done on a two-dimensional space, where one dimension is the left end mapping of the paired-end reads and other dimension is the size of the SV supported by the paired-end reads. In this method each paired-end read can be in only one cluster, and the reads with multiple mapping are resolved by selecting the cluster with highest number of paired-end read mappings in it [1].

1.5.3 Read-Depth Analysis

Read depth analysis are used for copy number variation discovery. The read-depth approaches have an implicit assumption that probability of sequencing each base pair obeys a Poisson distribution (the mean value of the distribution is the expected depth of coverage of sequencing).

Assuming we have the mapping location of all of the paired-end reads to the reference genome, then we define depth of coverage of a region as average number of reads which are mapped to that region [4]. It is obvious that the depth of coverage of a region is proportion to number of times that region appears in the donor genome. A significant divergence from the expected depth of coverage in a region in the reference genome is indicator of deletion or duplication of that region in donor genome in comparison to the reference genome. More precisely reduction of depth of coverage of that region indicates deletion of the region in the donor genome, while an increase in the depth of coverage in that region indicates duplication of the region in the donor genome. This method was used for discovery of recent segmental duplications in human genome [9] using old capillary sequencing. Finally, a slightly modified version of this approach is used for discovery of CNV using high-throughput sequencing technologies [4, 140].

1.5.4 Split-Read Analysis

The final approach used for structural variation discovery are based on split-read analysis. This approach is more useful for longer set of reads (like the reads produced by 454 sequencers, or the newer generation of Illumina sequencers). This framework was first proposed for capillary sequencing technologies (which were able to produce long reads) [106]. Split read approaches are able to detect deletions and (small) insertions with very high resolution (in theory split read approaches should be able to detect the exact breakpoint of these variation). The split read framework tries to identify the location in the reference genome where a read can be mapped to with gaps indicating insertion or deletion. Assuming a segment is deleted in donor genome, a read which spans the breakpoints will be mapped to the reference genome with the deleted segment being assigned to gap in alignment (the read will be broken in two segments). Such a split read signature indicates a deletion in donor genome. On the other hand assuming a small insertion in donor genome, where the read will be broken to three segments. The first and third segments will map to the reference genome and the middle segment will be mapped to gap. Please see figure 1.4 for a graphical illustration on how split-read analysis works.

Pindel [156]

Pinel [156] was the first method for structural variation discovery using high-throughput sequencing technologies using the split-read framework. Pindel uses a "pattern growth" method to predict the breakpoints of small to large scale deletions (10bp to 10kbp), and small size insertions (10 to 20bp) from the reads of length 36 bp produced by Illumina sequencer. Pindel algorithm only uses the mapping where the leftmost and/or the rightmost base of pattern P are mapped to the text. Pattern growth method is used to find minimum and maximum unique substring matching of a given pattern against the text.

Given a pattern P and text T, the minimum unique substring matching when the leftmost base of the pattern matches the text, is the smallest prefix of P which uniquely matches pattern T exactly (with zero edit distance), while the maximum unique substring matching is the largest prefix of P which uniquely matches the pattern T exactly. For the matching of the rightmost base pair of pattern P we are interested in the suffix of pattern P.

Large deletions: Pindel algorithm in the preprocessing step maps the paired-end reads to the reference genome using a mapper such as SSAHA2 [111]. Any paired-end read which only one end of it can be mapped to the reference genome (uniquely) and the other does not map to the reference genome is considered. Using the end that is mapped to the reference genome an approximate location for anchoring the other end of the paired-end read is calculated (knowing the minimum and maximum insert size of the clones). The steps pindel algorithm takes for detecting large deletions are as follows : 1) Pindel algorithm only considers paired-end reads which only one end of the read can map to the reference genome (Pindel also discards all the paired-end reads which the mapped end is not mapped uniquely) 2) Finds the minimum and maximum unique substring matchings from the 3' end of the unmapped read within two times of the insert size of the paired-end read using the pattern growth method 3) Finds the minimum and maximum unique substring matchings from the 5' end of the unmapped read with the range of maximum deletion allowed starting from the location of mapping found in first step 4) Tries to reconstruct the full read mapping by combining the locations and mappings found from the previous two steps.

1.6 Thesis Overview

In this thesis we will look at the algorithms for variation discovery (specifically structural variation discovery), with emphasis on repeats. In chapter 2 we will look at the problem of how to design a

resequencing experiment (using multiplex of regions of interest) such that the downstream analysis (e.g., variation discovery) becomes simpler; this problem is known as pooling problem. The results provided in this chapter are from our work published in [48]. The crux of the thesis is chapter 3, where the formulation for structural variation discovery under maximum parsimony assumption is introduced and approximation algorithms are provided. This chapter combines the algorithms and results published in [55], [58], [47] and [107]. Chapter 4 is an in depth look at the results on *Alu* insertion found in eight different individuals using the methods explained in chapter 3. The results in this chapter are from [56]. The chapter 5 of this dissertation is an extension of the methods explained in chapter 3 into multiple genomes, where we want to discover structural variation in multiple genomes simultaneously. The results provided in this chapter are published in [59]. Finally the conclusion and future works are provided in chapter 6.

Chapter 2 - Pooling Problem [48] : New generation sequencing technologies offer unique opportunities and challenges for re-sequencing studies. In chapter two of this thesis we focus on algorithms for re-sequencing experiment design using high-throughput technologies (e.g., Illumina/Solexa technology), based on bacterial artificial chromosome (BAC) clones. In these specific experiments, approximate coordinates of the BACs on a reference genome are known, and fine scale differences (SNV, indels or SVs) between the BAC sequences and the reference are of interest. The high-throughput characteristics of the sequencing technology make it possible to multiplex BAC sequencing experiments by pooling BACs for a cost-effective operation. Existence of repetition in different BACs pooled and sequenced together makes the problem of variation discovery (such as structural variation discovery) much harder through an increase in the frequency of multiple mapping of paired-end reads produced. The experimental design strategy we develop offers combinatorial solutions based on approximation algorithms for the well known *max n-cut* problem and the related *max n-section* problem on hypergraphs. Our algorithms, when applied to a number of sample cases, give more than a 2-fold performance improvement over random partitioning. Finally note that although the pooling problem defined and algorithms developed in this chapter are mainly being utilized in the context of BAC re-sequencing, they can easily be applied to other re-sequencing experiments. For instance exome sequencing projects can benefit from the algorithms developed in this chapter.

Chapter 3 - Structural Variation Discovery and Maximum Parsimony [55, 47, 58] : Recent studies show that along with single nucleotide polymorphisms and small indels, larger structural variants among human individuals are common. The Human Genome Structural Variation Project

aims to identify and classify deletions, insertions, and inversions ($> 5Kbp$) in a small number of normal individuals with a fosmid based paired-end sequencing approach using traditional sequencing technologies. The realization of new ultra high throughput sequencing platforms now makes it feasible to detect the full spectrum of genomic variation among many individual genomes, including cancer patients and others suffering from diseases of genomic origin. In addition high throughput sequencing technologies make it possible to extend the scope of structural variation studies to a point previously unimaginable as exemplified by the 1000 Genomes Project. In this chapter we consider the structural variation discovery problem, when we are dealing with *multiple mapping* for paired-end reads. We give combinatorial formulations for the structural variation detection between a reference genome and a high-throughput based, paired-end, whole genome shotgun sequenced individual based on the maximum parsimony principle. For the above formulation we develop an approximation algorithm with an approximation factor of $\log n$, and we denoted this method as *VariationHunter* [55] (n is number of paired-end reads indicating a structural variation). Then, we extend the maximum parsimony proposed framework to consider the haploid or diploid nature of the genome being processed. We introduce the "conflict resolution" improvements to our earlier combinatorial SV detection algorithm (VariationHunter) for solving the extended problem. The method we develop for the extended framework is known as VariationHunter-CR [58] (VariationHunter with Conflict Resolution). Our methods are able to find different types of structural variations, including small insertions, deletions, inversion, and transpositions (mobile element insertions). Note that VariationHunter and its extensions have been used successfully in large scale projects, such as the 1000 Genomes Project [107, 1] and Gorilla Genomes Project [146]. Finally an extension of the maximum parsimony framework is used for finding *novel* insertions in the donor genome [47].

Chapter 4 - ***Alu*** **insertion discovery [56] :** Human genomes are now being rapidly sequenced, but not all forms of genetic variation are routinely characterized. In this chapter, we focus on *Alu* retrotransposition events and seek to characterize differences in the pattern of mobile insertion between individuals based on the analysis of eight human genomes sequenced using next-generation sequencing. Applying a read-pair analysis algorithm (VariationHunter as explained in chapter 3), we discover 4342 *Alu* insertions not found in the human reference genome and show that 98% of a selected subset (63/64) experimentally validate. Of these new insertions, 89% correspond to *AluY* elements, suggesting that they arose by retrotransposition. Eighty percent of the *Alu* insertions have not been previously reported, and more novel events were detected in Africans when compared with non-African samples (76% vs. 69%).

Chapter 5 - Simultaneous structural variation discovery in multiple genome [59] : 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, ethnicity, or a group sharing a particular phenotype). The conventional approach for a comparative genome variation study involves two key steps: i) each high throughput sequenced genome is compared with a reference genome and its (structural) differences are identified, and ii) the list of structural variants in each genome are compared against each other. We propose a paradigm shift from this two-step approach to a novel one in which all genomes are compared simultaneously with the reference genome for obtaining much higher accuracy in structural variation detection. For this purpose, we introduce the maximum parsimony-based simultaneous structural variation discovery problem for a set of high throughput sequenced genomes and provide efficient algorithms to solve it. We compared the proposed framework to the conventional framework, on the genomes of the Yoruban mother-father-child trio (sequenced by the Illumina platform). We observed that the conventional framework predicts an unexpectedly high number of de novo variations in the child in comparison to the parents and misses some of the known variations. Our proposed framework, on the other hand, not only significantly reduces the number of incorrectly predicted de novo variations but also predicts more of the known (true) variations.

Chapter 2

Optimal pooling for genome re-sequencing

After decades of research effort, the cost of sequencing an individual human genome via Sanger sequencing [130, 128, 129] has now been reduced to the order of 10-million dollars for a 10-fold coverage and require 10000 instrument days (one year with 30 instruments) [10]. In order to perform fast and cost effective genome sequence comparisons for many biological and medical applications, the cost needs to be further reduced to the order of a few thousand dollars (i.e. a few dollars per megabase) and the time frame of sequencing must be reduced to a few days per instrument. For example, the Illumina/Solexa sequencing-by-synthesis technology of Illumina offers about 100 to 1000-fold improvements over Sanger sequencing in both cost and throughput. Similarly, the pyrosequencing technology of 454 Life Sciences [94] delivers massive parallelization of the sequencing process by the use of microchip sensors, improving the speed of Sanger sequencing by a factor of a few hundred. In addition to these, there are other commercial products from Applied Biosystems, Helicos Biosciences and Visigen Biotechnology, among others that 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 around 1000bps per read, the read lengths of new technologies range from 36 to 100 (e.g. Illumina/Solexa) to few hundred base pairs (e.g. 454) and as such, they are suitable for re-sequencing studies. Although in practice most sequencing technologies can produce longer, as well as paired end reads, but for rest of this chapter

we concentrate on the problem of resequencing of genomes with short single end reads.

In this chapter we look at problem of how to design resequencing experiments so to decrease the complexity of 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.

We assume that the donor genome's regions of interest are available as segments of DNA and for each of these segments we know (approximately) which part of genome they come from. These segments of genome can be BACs, fosmids or small fragments containing exons of interests (from exome sequencing experiments). For this chapter we focus on BAC re-sequencing experiments as the test platform of our algorithm, however it can as easily be applied to exome sequencing experiments or fosmid resequencing. Note that one can a priori determine the genomic neighbourhood the BAC or a fosmid is coming from, thus having the reference sequence provide an essential backbone. This can be achieved, for instance, by using the results of fingerprinting or end sequencing experiments. A BAC or fosmid library-based sequencing enables directed sequencing studies possible, where the interest is not on whole genome (re)sequencing, but on investigating a region of interest. Note that in exome sequencing projects regions of interests are coding regions of genome.

Illumina sequencing experiments are run on a flow cell with eight lanes, each yielding in the order of 10^8bps of sequence per run. 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/Solexa 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 *entanglement* 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 DNA fragment assembly [123, 115, 114], especially those for short reads [25, 26, 151] all suffer from the presence of repeats within the genome region to be assembled [94]. 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 [141]) 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/Solexa 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. We show our algorithms are quite efficient in practice.

In this chapter we define the pooling problem and apply our algorithms in context of BAC resequencing. However the problem defined and the algorithm devised is much more general than only BAC resequencing experiment design. For instance we can apply the same concept in the resequencing of fosmid clones. As in recent study [77] 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). Assuming that we can approximately predict the region of reference genome where each fosmid clone maps to, our algorithm can optimize the pooling of the fosmids so that it reduces the overall ambiguity in read mapping for downstream analysis.

Another very promising application of our algorithm is in exome sequencing. In exome sequencing the goal is to sequence the coding regions of donor genome. One main part of most exome sequencing protocols is genome partitioning, which is to enrich the sequence library (donor genome) for specific regions (i.e. exons). There are many methods for capturing exons from the donor genome, like utilizing multiplex PCR, capture-by-hybridization or capture-by-circulation [143]. The captured exons are sequencing using high-throughput technologies for further analysis for variation discovery on the captured exons. The decision on which set of exons should be pooled together in the capture and sequencing phase such that to reduce the ambiguity in mapping phase can have significant effect in final results. The problem and methods proposed in this chapter can easily be applied for these scenarios.

2.1 Problem definition and general algorithmic approach

Given a set of genome fragments from known specific genomic regions of interest, our ultimate goal is to construct the sequence of each fragment (which can be BACs, fosmid or captured exons in an exome experiment) using the results of optimally designed Illumina sequencing experiments. Consider a set of m segments of genome (for instance BACs) sequenced with a read length of k (typically 25 to 100bps); 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 segments (for instance 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, mark that our focus is on re-sequencing studies, and if we have even a crude idea on the genomic coordinates of the segments, we can approximate that missing information by using the reference sequence. For instance for BACs resequencing using technologies like finger-printing can provide us these coordinates, while for exome sequencing, we can modify the capturing phase of exons such that each pool of exons is distinguishable from another set.

One other hurdle in designing a globally optimal experiment is the rapid proliferation of number of possible configurations. Since it is infeasible to make an exhaustive search for the globally optimal configuration, we propose an algorithmic approach to guide us to an "approximately" optimal setup. Of course for a exome sequencing experiments, which we are dealing with around $200,000$ exons (fragments) we can pool them to tens of pools (we can use barcoding technology to distinguish segments/exons of each pool from others).

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.

2.1.1 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 [126]. 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 [126], 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 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 [126] 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. □

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 [42] 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 simple 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 [42] 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 $|S| \leq d$).

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

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.

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:

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

.

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,

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

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 ([108]).[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

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

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.

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 [97, 125]. 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 [80] 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

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

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

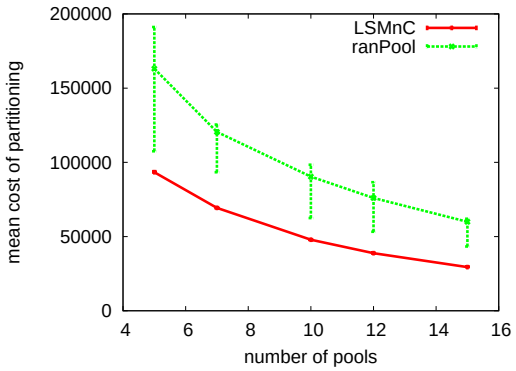
We first compare the performance of our local search methods LSMnC/LSMnS with random partitions (denoted *ranPool*). Although these methods were designed to minimize the cost of pooling with respect to the cost function $f(C_{i,b}) = \binom{C_{i,b}}{2}$, we report their performance with respect to the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

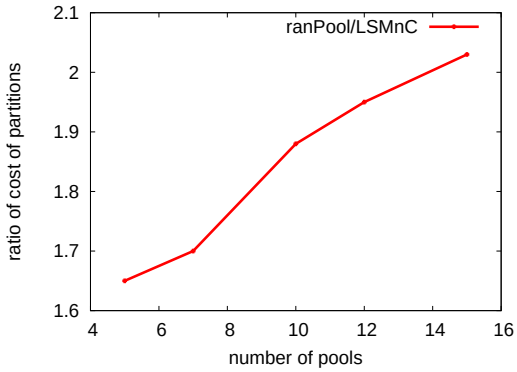
In figures 2.1(a) and 2.2(a) we give the distribution (the mean value as well as the highest and lowest 25%) of the cost obtained by 5000 independent runs of LSMnC/LSMnS and ranPool methods on the *lymphoma* and the *synthetic* data sets. The figures show how the cost changes with respect to the increasing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

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).

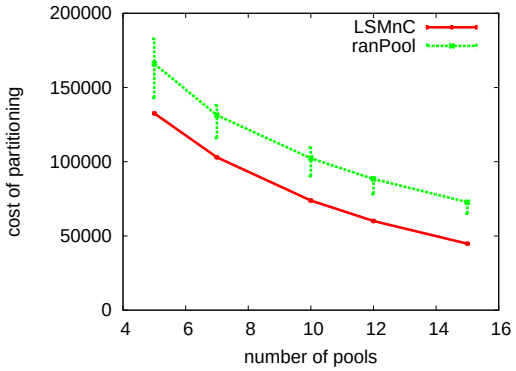


(a) The change of cost with respect to the number of pools.

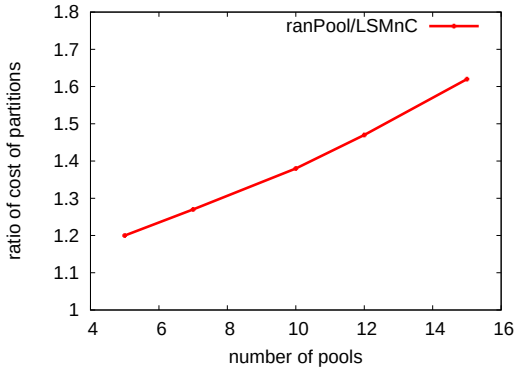


(b) The ratio between the costs of random partitioning and the LSMnC/LSMnS algorithms.

Figure 2.1: 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.



(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.

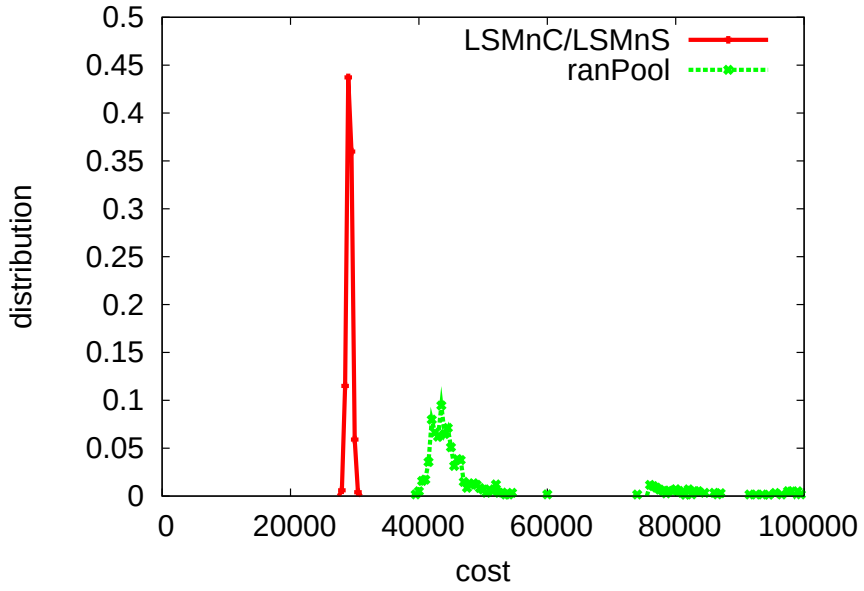


Figure 2.3: The distribution of cost obtained by ranPool and LSMnC/LSMnS for $n = 15$ on the lymphoma data set after 5000 independent runs.

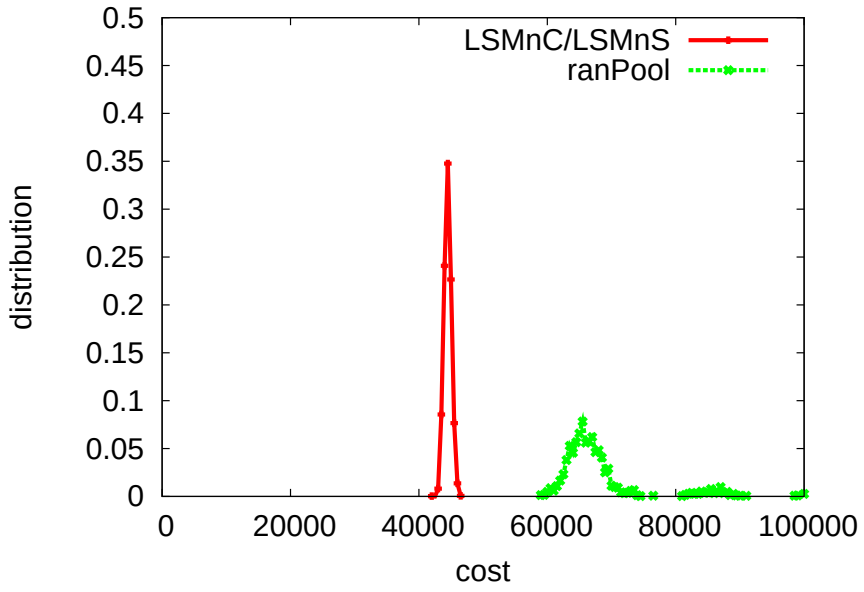


Figure 2.4: The distribution of cost obtained by ranPool and LSMnC/LSMnS for $n = 15$ on the synthetic data set after 5000 independent runs.

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.

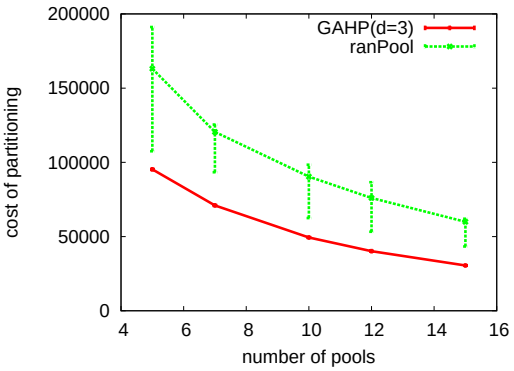
Our second set of algorithms, GAHP/GABHP are designed to "minimize" the cost with respect to the second cost function. They are flexible in the sense that one can set up the value of d as desired; for $d = n$, the optimal solution to the hypergraph partitioning indeed minimizes the cost function $f(C_{i,b}) = C_{i,b} - 1$. 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).

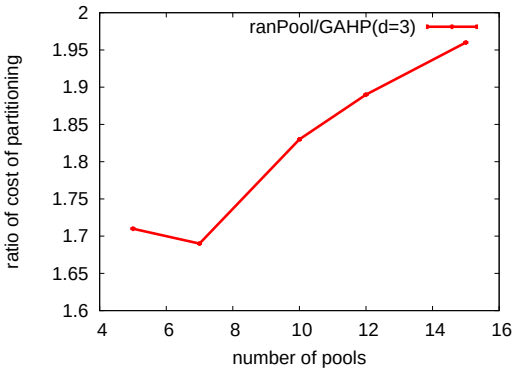
We also investigate the effect of using hypergraphs with $d = 3$ over the use of ordinary graphs with $d = 2$ on the GAHP/GABHP methods. We again report the results of 5000 independent trials on both data sets in figures 2.7(a) and 2.7(b).

The performance improvement achieved by increasing d from 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

[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 $n = 15$.

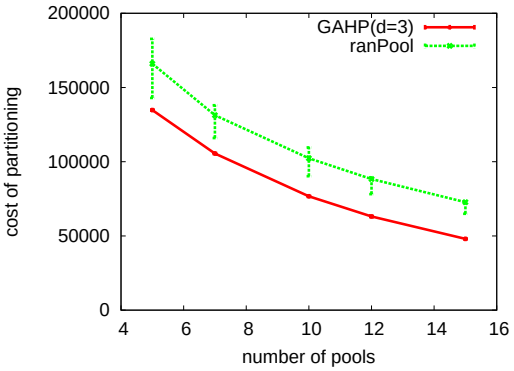


(a) The change of cost with respect to the number of pools.

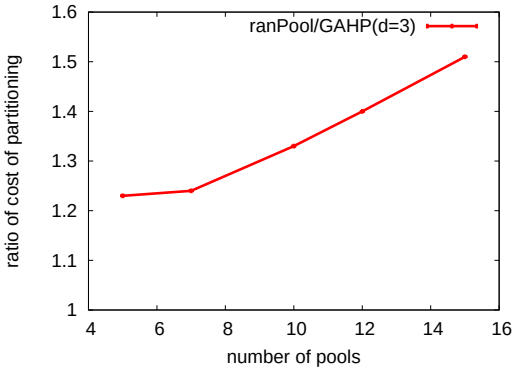


(b) The ratio between the costs of random partitioning and the GAHP/GABHP algorithms.

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.

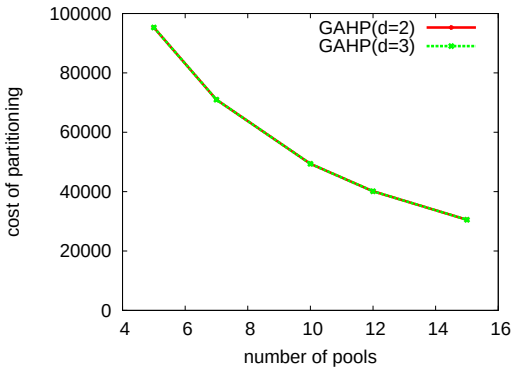


(a) The change of cost with respect to the number of pools.

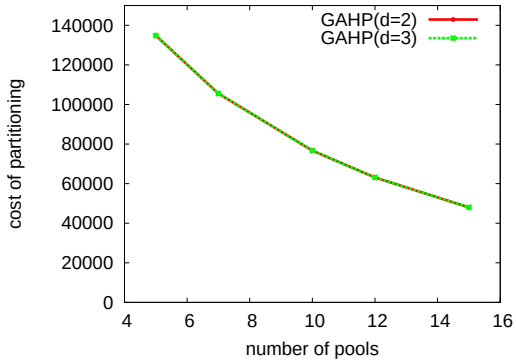


(b) The ratio between the costs of random partitioning and the GAHP/GABHP algorithms.

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.



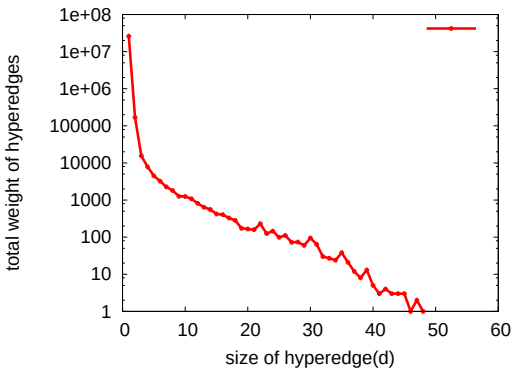
(a) Comparing the performance of GAHP/GABHP method for $d = 2$ and $d = 3$ on the lymphoma data set.



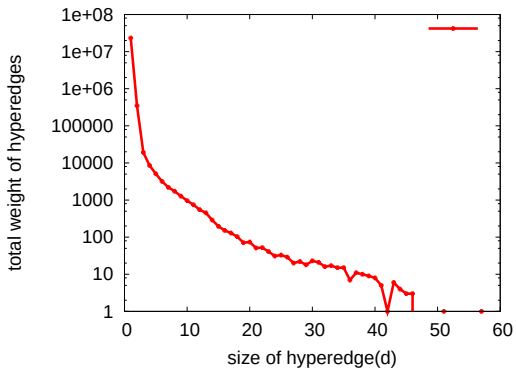
(b) Comparing the performance of GAHP/GABHP method for $d = 2$ and $d = 3$ on the synthetic data set.

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

$100 - 200$ 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.



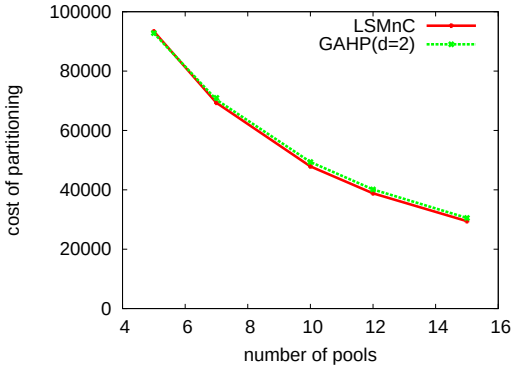
(a) Distribution of hyperedge weights on the lymphoma data set.



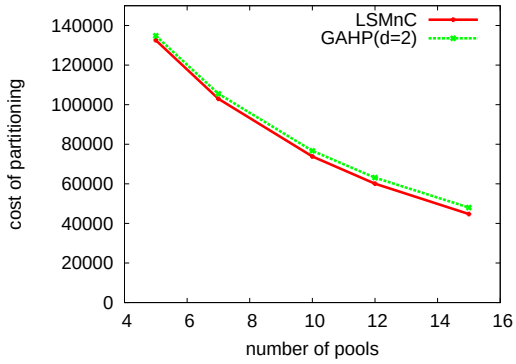
(b) Distribution of hyperedge weights on the synthetic data set.

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

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 $d = 2$).



(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.

2.4 Other Applications

In this chapter we consider pooling problem in the context of BAC resequencing. However, the problem introduced and approximation algorithms provided can be used in improving the results in other resequencing experiments. Two examples include fosmid resequencing and exome sequencing.

2.4.1 Fosmid clone resequencing

In recent publication by Kitzman et al [77], the authors first constructed a fosmid library from a Gujarati individual and split the libraries into 115 pools, each pool was barcoded, and then all pools were sequenced together. With the assumption that for each fosmid we can approximately locate where in reference genome it maps to, in theory we can optimize the pooling step of the in [77] to reduce total number of ambiguous mappings in each pool.

2.4.2 Exome sequencing

One of the more exciting applications of our pooling algorithm can be in exome sequencing. In exome sequencing the goal is to discover the variations on the coding segments of donor genomes using high-throughput sequencing technologies. For achieving such a goal a specialized genome partitioning method is used to select the fragments of exons from donor genome (using Multiplex PCR, capture by circularization, or hybridization methods), and the fragments selected is sequenced. One way to use our pooling algorithms to improve the final result of exome sequencing experiments, is to decide which set of exons should be captured and sequenced together. The goal is to reduce the complexity of downstream analysis (i.e. amount of ambiguous mapping in each pool of captured exons is minimized). For instance in the publication by Hodges et al [52] they have captured more than $200,000$ coding exons using eight different microarrays. The process of how to pool these coding exons between eight different microarrays, such that the downstream analysis is made simpler can be answered by the pooling algorithms we provided.

Chapter 3

Structural Variation Discovery Via Maximum Parsimony

Recent introduction of the next-generation sequencing technologies has changed how genomics research is conducted significantly [93]. High-throughput, low-cost sequencing technologies such as *pyrosequencing* (454), *sequencing-by-synthesis* (Illumina and Helicos), and *sequencing-by-ligation* (SOLiD) methods produce shorter reads than the traditional capillary sequencing, but they also increase the redundancy by $10-100$-fold, or more [93, 135]. With the arrival of these new sequencing technologies, along with the capability of sequencing paired-ends (or *mate-pairs*) of a clone insert that follows a tight length distribution [147, 119, 144, 34, 79, 73, 84, 12], it is becoming feasible to perform detailed and comprehensive genome variation and rearrangement studies.

The genetic variation among human individuals has been traditionally analyzed at the single nucleotide polymorphism (SNP) level, as demonstrated by the HapMap Project [65, 66] where the genomes of 270 individuals were systematically genotyped for 3.1 million SNPs. However, human genetic variation extends beyond SNPs. The Human Genome Structural Variation Project [35] has been initiated to identify and catalogue structural variation. In the broadest sense, structural variation (SV) can be defined as the genomic changes among individuals that are not single nucleotide variants [144, 35]. These include insertions, deletions, duplications, inversions and translocations [134, 41].

End sequence profiling (ESP) was first presented in [147, 119] to discover structural variation

events using the bacterial artificial chromosome (BAC) end sequences to map structural rearrangements in cancer cell line genomes; and it was used in [144] to systematically discover structural variants in the genome of a human individual. Several other genome wide studies [79, 63, 133, 121, 29] demonstrated that structural variation among normal individuals is common and ubiquitous. More recently, Kidd et al. [73] detected, experimentally validated, and sequenced structural variation from 8 different individuals. ESP method was also utilized by Dew et al. [34] to evaluate and compare assemblies and detect assembly breakpoints.

As the promise of these next generation sequencing (NGS) technologies become reality with the publication of the first three human genomes sequenced with NGS platforms [152, 18, 150], the sequencing of more than $1,000$ individuals (http://www.1000genomes.org), computational methods for analyzing and managing the massive numbers of the short read pairs produced by these platforms are urgently needed to effectively detect single-nucleotide polymorphisms, structural, and copy-number variants [116]. Since most structural variation events are enriched in the duplicated regions [73, 35], the algorithms must also be able to discover variation in the repetitive regions of the human genome.

Detection of structural variants in the human genome using next-generation sequencing technologies was first presented in [79]. In this study, paired-end sequences generated with the 454 platform were employed to detect structural variants in two human individuals, however the same algorithms and heuristics designed for capillary based sequencing presented in [144] were used, and no further optimizations for NGS were introduced. Campbell et al. employed Illumina sequencing to discover genome rearrangements in cancer cell lines, however they considered one "best" paired mapping location per insert [23], by the use of the alignment tool MAQ [88], thus not utilizing the full information produced by high-throughput sequencing methods. In the first study on the genome sequenced with a NGS platform (Illumina) that produced paired-end sequences [18], Bentley et al. also detected structural variants using the same methods and unique map locations of the sequenced reads.

Prior to publication of our method, a probabilistic method by Lee et al. [84] was presented for detecting structural variation. In this 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. In theory the method in [84] can be applied to data generated by new sequencing technologies, however the experiments presented in [84] was based on capillary sequencing[87]. In

another study Bashir et al. [12] presented a computational framework to evaluate the use of paired-end sequence analysis detect genome rearrangements and fusion genes in cancer [12]; note that no NGS data was utilized in this study due to lack of availability of sequences at the time of publication.

In this chapter we present combinatorial algorithms for structural variation detection using the NGS methods. We define two alternative formulations for the problem of computationally predicting the structural variation between a reference genome sequence (i.e. human genome assembly) and a set of paired-end reads from a whole genome shotgun (WGS) sequence library obtained via an NGS method from an individual genome sequence. The first formulation, which we call *Maximum Parsimony Structural Variation Problem* (MPSV), turns out to be NP-hard; we give a $O(\log n)$ approximation algorithm to solve this problem in polynomial time. The method proposed to solve MPSV problem is called *VariationHunter* method (in short *VariationHunter*). VariationHunter algorithm capable to discover structural variations of type small insertion, deletion and inversion was published in [55]. The extension of VariationHunter to discovery copy/transposition events was published in [58] (in this chapter we also present a weighted extension of MPSV and weighted VariationHunter).

The second formulation tries to avoid SV predictions which are in "conflict" with each other. It extends the maximum parsimony formulation to non-conflicting maximum parsimony. The algorithm developed for the second formulation is known as VariationHunter-CR [58]

3.1 Insertion, Deletion, Inversion and Transposition Events

3.1.1 Notations and Definition

A general method for using *paired-end* long reads to detect structural variants between *new donor genome* and *reference genome* was first introduced in [147, 144]. This general strategy is based on aligning the paired-end sequenced reads to the reference genome and observing significant differences between the distance of *mate pairs*[1] when mapped to reference genome and their expected distance - which indicates a *deletion* or an *insertion* event. Furthermore, one can also deduce inversion events: if one of the two ends of a pair has a "wrong" orientation, this is likely a result of an *inversion* (see [144]). In case, the two ends are *everted*, i.e. both ends of a paired-end have reverse orientation (with respect to each other), but their order is preserved in the reference genome, it is likely to have a *tandem repeat*. Finally, paired-end reads mapping to two different chromosomes are

[1] Mate pairs refer to the two ends of a paired-end read.

likely to be a result of a *trans-chromosomal* event (see Figure 3.1).

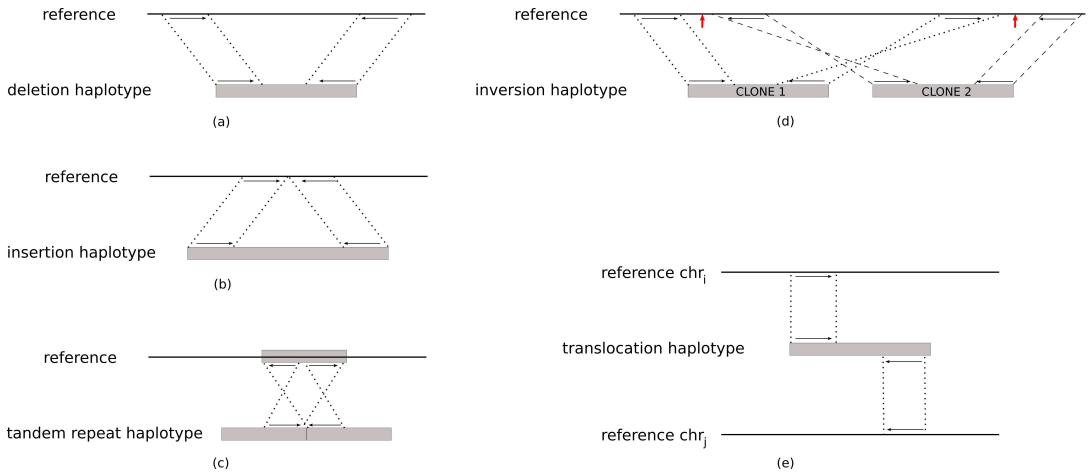


Figure 3.1: Types of structural variation that can be detected with paired-end sequences: mapped span of paired-end reads appear larger than the expected insert length if there is a (a) deletion and smaller in an (b) insertion haplotype. Disagreement between the mapping orientations and the sequencing library specifications might either report a (c) tandem repeat or an (d) inversion. Also, note that in the case of inversions $CLONE_1$ and $CLONE_2$ predict two different inversion breakpoints (shown with arrows), but by examining the map locations and orientations, one can deduce that both clones predict the same inversion, and both breakpoints of the inversion event can be recovered. If the paired-end reads align confidently on different chromosomes, a (e) translocation event is reported. In this figure, we assumed the expected end-sequence orientation properties in capillary based sequencing and Illumina platforms. Figure from [55]

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 [144, 79, 84] assume that for all paired-ends, $InsSize$ is in some range $[\Delta_{min}, \Delta_{max}]$ which can be calculated as described in [144].

An alignment of a paired-end read to reference genome is called *concordant* [144], if the distance between 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 [144] and later used in [79, 84], is called a *discordant* paired-end read (which indicates a possibility of a structural

variation).

Let the set of discordant paired-end reads be represented as $DisCor = \{pe_1, pe_2, \cdots, pe_n\}$. Each of these discordant mate pairs can have multiple (pairs of) locations in genome that they can be aligned to with high sequence similarity (e.g. $\geq 90\%$ [2]), can be represented by $Align(pe_i) = \{a_1 pe_i, a_2 pe_i, \cdots, a_j pe_i\}$. We also define $Align^{\leftarrow}(a_j pe_i) = pe_i$.

Note that each alignment location in the reference genome (as mentioned above, all alignments of paired-end reads to the reference genome require a sequence similarity above 90%), $a_j pe_i$, includes a tuple of a pair of loci in genome and orientation of the mapping. More formally,

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

where the pair $(L_\ell(a_j pe_i), L_r(a_j pe_i))$ represents the map location (i.e. both start and end loci) of the left end-read of $a_j pe_i$, $(R_\ell(a_j pe_i), R_r(a_j pe_i))$ is the mapping location of the right end-read of $a_j pe_i$, and $or(a_j pe_i)$ represents the map orientation of both ends. Note that $or(a_j pe_i) \in \{+-, ++, --, -+\}$ ($or(a_j pe_i) = +-$ is the orientation representing no inversion, $or(a_j pe_i) = ++$ shows an inversion event where right mate pair is in the inverted region and $or(a_j pe_i) = --$ represents the fact that left mate pair is in the inverted region).

Our algorithm(s) will obtain a unique alignment $Map(pe_i)$ from the set $Align(pe_j)$ for each paired-end read pe_i. We denote by $Map_{corr}(pe_i)$, the "correct" location for the paired-end read pe_i. The goal of our algorithm(s) is to pick $Map(pe_i) = Map_{corr}(pe_i)$.

A set of discordant paired-end read mappings is called a "valid" cluster $VClu_i$ if there exists a structural variation which all of the mappings in the set $VClu_i$ can "support" it (i.e. the existence of one structural variation can explain the discordant paired-end read mappings in the cluster). We will define the concept of valid cluster in more detail in this chapter. An alignment of paired-end read, such as ape, is said to be "materialized" by the algorithm if it maps the paired-end read, $Align^{\leftarrow}(ape)$, to alignment ape. A valid cluster $VClu_i = \{a_{i'_1} pe_{i_1}, a_{i'_2} pe_{i_2}, \cdots, a_{i'_\ell} pe_{i_\ell}\}$ is said to be "materialized" (by the algorithm) if for each j, $Map(pe_{i_j}) = a_{i'_j} pe_{i_j}$. We denote materialized clusters as $MClu_j$.

[2] This an arbitrary cutoff; using higher cutoff value makes the problem easier, however we might miss some real structural variants.

3.1.2 Structural Variation Detection based on Maximum Parsimony

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. The minimum number of SVs implied by the mappings is the most parsimonious one under the implicit assumption that all SVs are equally likely. Note that minimizing the SVs also will imply that the average number of paired-end reads supporting an SV is maximized - the two goals are equivalent. Note that for the MPSV problem we provide an algorithm with an approximation guarantee.

More formally, MPSV problem asks to compute the minimum number of materialized clusters (sets) $MClu_i$ given a set of $DisCor$ paired-end reads and a set alignment locations ($Align(pe_i)$) for each paired-end read pe_i such that

$$\begin{aligned} \{pe | pe = Align^{\leftarrow}(ape) : ape \in \cup_{\forall i} MClu_i\} &= DisCor \\ \forall ape_1, ape_2 \in \cup_{\forall i} MClu_i : Align^{\leftarrow}(ape_1) = Align^{\leftarrow}(ape_2) &\implies ape_1 = ape_2 \end{aligned} \quad (3.1)$$

The MPSV problem can be further constrained as per [144, 79, 84] so that each materialized cluster includes at least two reads. The problem can also be generalized, in a way that each structural variation has an associated cost, which may be based on the sequence similarity in the alignments, the number of pair-end reads supporting it, and length of structural variation. We have proven that MPSV problem is NP-hard using a simple reduction from set cover problem .

Theorem 1. *MPSV problem is NP-hard.*

Proof. Reduction is from the set cover problem [72]. Given a set $U = \{e_1, \ldots, e_n\}$ and $S = \{S_1, S_2, \cdots, S_k\}$, a collection of subsets of U, the set cover problem asks to find the minimum number of sets in S whose union include all $e_i \in U$. The reduction from an instance of a set cover problem to the MPSV problem is as follows:

1. Set $DisCor = U$, that is, for each e_i generate a paired-end read pe_i.
2. For each set S_i set an interval (L_{S_i}, R_{S_i}), which does not overlap with any other such interval.
3. Finally, set $Align(pe_i) = \{(L_{s_j}, R_{s_j}) | \forall S_j : e_i \in S_j\}$.

Clearly, the two problems are equivalent and a subset S' of S is a minimum size set cover of S iff the set of intervals corresponding to S' includes the minimum number of intervals to which each paired-end read pe_i can be mapped to. □

3.1.3 $O(\log n)$ approximation for MPSV

Before proceeding with the approximation algorithm for the MPSV problem we have to introduce the notion of a **maximal valid cluster**. A "maximal valid cluster" is a valid cluster for which no valid superset exists. As defined in section 3.1.1, a set of discordant paired-end read alignments that support the same potential SV event is called valid cluster and denoted by $VClu_i = \{a_{i'_1}pe_{i'_1}, a_{i'_2}pe_{i'_2}, \cdots, a_{i'_\ell}pe_{i'_\ell}\}$ (i.e. there is in theory a structural variation possible to justify the discordant paired-end read mappings in the valid cluster). Thus, a maximal valid cluster is a valid cluster which no additional paired-end read mapping can be added such that it remains valid [3].

Although the number of valid clusters can be exponential, however number of total maximal valid clusters is polynomial and can be produced in polynomial time. We will provide an efficient method to find all the maximal valid clusters for each type of structural variation in section 3.1.4.

Given a universe set $U = \{e_1, \ldots, e_n\}$ and a collection of subsets of U, $S = \{S_1, S_2, \cdots, S_m\}$, the set cover problem asks to find the smallest subset of S whose union covers each $e_i \in U$. The greedy algorithm, which at each iteration picks up the set that includes the maximum number of uncovered elements of U until all elements of U are covered, provides an $O(\log n)$ approximation to the optimal solution [145]. Interestingly enough this simple algorithm implies an approximation factor of $O(\log n)$ for the MPSV problem after the following modification: in each iteration of the algorithm pick up the maximal valid cluster with the maximum number of uncovered paired-end reads (the proof for the approximation factor trivially follows the proof for the set cover problem [145]). We have named this algorithm *VariationHunter-Set Cover* (in short *VariationHunter*), because of its use of original set cover algorithm. Note that we have used a modified VariationHunter for discovery of gene fusions using RNA-Seq data [99] (the algorithm for gene fusion discovery is called *deFuse*).

[3] Note that the concept of maximal valid cluster was first introduced in [55], and (later) independently in [137]. In [137] the name *maximal intersecting breakpoint regions* was giving to this concept.

Weighted MPSV

We extend the MPSV problem such that we can assign weights to each structural variation based on type and length of the event. For each potential structural variation we have a weight function W (in unweighted MPSV problem this is equal to 1 for every potential structural variation).

In addition we can assume for each paired-end read pe_i different mappings have different weights. Some mappings might suggest more SNPs and indels (i.e. have higher edit distance) in comparison to other mappings. Lets assume that for each possible mapping of paired-end read pe_i (such as $a_j pe_i$) we have a weight $w(a_j pe_i)$ which is an increasing function on edit distance of the mapping ($w(a_j pe_i) = 0$ if $a_j pe_i$ is a perfect mapping, i.e. zero edit distance). The weighted maximum parsimony structural variation problem asks to assign all discordant paired-end reads to mappings such that summation of weights of structural variations (W) predicted and weights of alignment selected (w) is minimized.

More formally weighted MPSV tries to find a set of materialized clusters $MClu_i$ given the set of $DisCor$ paired-end reads and the set of alignment locations (i.e. $Align(pe_i)$ for each paired-end read $pe_i \in DisCor$) to minimize objective function

$$\sum_{\forall MClu_i} W(MClu_i) + \sum_{\forall ape \in \cup_{\forall i} MClu_i} w(ape)$$

satisfying the same constraints as MPSV (see conditions 3.1.2).

We also provide a $O(\log n)$ approximation for the weighted MPSV problem, which we call *weighted VariationHunter*. We use the greedy algorithm provided for set cover for solving this problem (with slight modifications) as follow:

Assume that we have all the maximal valid clusters (i.e. $VClu_i$) provided (as explained in section 3.1.4). For each maximal valid cluster $VClu_i = \{a_{i'_1} pe_{i'_1}, a_{i'_2} pe_{i'_2}, \cdots, a_{i'_\ell} pe_{i'_\ell}\}$ we consider each of its subsets as a distinct set denoted by $S_{i_j} \subseteq VClu_i$ (when $j \in \{1, 2, \cdots, 2^\ell - 1\}$). To each set S_{i_j} we assign the cost $W(VClu_i) + \sum_{\forall ape \in S_{i_j}} w(ape)$, and try to find a subcollection of these sets with minimum total cost to cover all the elements (i.e. paired-end reads).

We utilize the greedy algorithm for weighted set cover [145] to pick the sets to cover all the elements with approximation factor of $O(\log n)$. Note that the greedy method for weighted set cover in each iteration picks the set with least "cost-effectiveness". The cost-effectiveness of each set (e.g. S_{i_j}) in each iteration is defined as total cost of the set S_{i_j} divided by number of (uncovered) elements (i.e. paired-end reads) being covered by set S_{i_j} for the first time. The only remaining

concern is that total number of sets S_{i_j} considered are exponential. Thus, we need to show how we can pick the most cost-effective set at each iteration, among all subsets of maximal valid clusters in polynomial time.

For each maximal valid cluster $VClu_i$ (with cardinality of ℓ paired-end read mappings) instead of calculating the cost-effectiveness of all its subsets in each iteration (i.e. $2^\ell - 1$ subsets), it is sufficient to calculate the cost-effectiveness for ℓ subsets of $VClu_i$ at each iteration.

Note that the most cost effective subset of a valid cluster $VClu_i$ which covers k new elements is the set of k uncovered elements from $VClu_i$ with least amount of total weight. To find such a subset in polynomial time, we sort (in ascending order) the elements in each cluster (i.e. paired-end read mappings) based on their weight (w) and pick the top k uncovered elements. We can do this for every k ($1 \leq k \leq \ell$), to pick the most cost-effective subset of $VClu_i$. This producer would eliminate the need for exploring exponential different subsets and guaranties an approximation of $O(\log n)$ for the weighted MPSV proposed.

3.1.4 Maximal Valid Clusters

In this section we will provide algorithms for finding all maximal valid clusters for structural variation events of type insertion, deletion, inversion and transposition (e.g. mobile element insertion) efficiently. First we give an optimal algorithm which is able to find all maximal intersecting intervals given a set of intervals. We utilize this algorithm as a subroutine in finding maximal valid clusters.

Maximal intersecting intervals

Given a set of intervals $I = \{[I_{1_L}, I_{1_R}], [I_{2_L}, I_{2_R}], \cdots, [I_{m_L}, I_{m_R}]\}$ with cardinality m, we describe an optimal method to find all the maximal intersecting intervals of it. A set of intervals $I' \subseteq I$ is "intersecting" if there exists a common intersection for all the intervals in the set I' (i.e $\exists x \in \mathbb{R}, \forall J \in I' : J_L < x < J_R$). A maximal set of intersecting intervals from set I is an intersecting set of intervals where no superset of it is also an intersecting set of intervals. Here we provide an algorithm which finds all maximal intersecting intervals of set I in optimal time.

We first sort all end-points of m intervals ($2m$ coordinates) 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 that is the left end-point of an interval, we insert the interval to a minimum heap data structure, denoted as $heap$. The priority value of the $heap$ is the right end-point of the inserted interval. After each insertion of a new interval to $heap$, we set a flag $newIns = true$.

- If we observe a point that is the right end-point of an interval we take these two steps. 1) We check the flag $newIns$ and if it is set to *true* we output all the elements in the $heap$ as one maximal intersecting interval and set $newIns = false$. 2) We remove the interval from $heap$ since it is guaranteed that the value of the right end-point of the interval removed from the heap is the same as the right end-point of the interval reached in scanned list L. We continue removing intervals from $heap$ till the priority value of the head element of the heap changes.

The above algorithm outputs all of the maximal intersecting intervals of input set I. We denote this algorithm as $MInIn$ (short for *Maximal INtersecting INtervals*).

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

(Small) Insertion

We represent a structural variation of type insertion by event $SV_{Ins}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$. The event $SV_{Ins}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$ represents an insertion between loci Loc_L and Loc_R of the reference genome, with the length of insertion being in range of Ran_{min} and Ran_{max}. A set of paired-end read mappings Clu can support the same insertion if and only if two conditions are met. First, there exists a locus in genome were all of the paired-end read mappings in the cluster Clu span it. Second, there exists a positive lengths of insertion, Ψ, supported by all the paired-end read mappings in the cluster Clu.

Let paired-end read mapping $a_j pe_i$ support an insertion event (i.e. $R_\ell(a_j pe_i) - L_r(a_j pe_i) < \Delta_{min}$). It is not hard to see that the lengths of insertion, $\Psi(a_j pe_i)$ (supported by the paired-end read mapping $a_j pe_i$) is in the range provided in equation 3.2.

$$\Delta_{min} + L_r(a_j pe_i) - R_\ell(a_j pe_i) < \Psi(a_j pe_i) < \Delta_{max} + L_r(a_j pe_i) - R_\ell(a_j pe_i) \quad (3.2)$$

Thus we can assume $\Psi(a_j pe_i) = [\Psi_{min}(a_j pe_i), \Psi_{max}(a_j pe_i)]$, where $\Psi_{min}(a_j pe_i) = \Delta_{min} + L_r(a_j pe_i) - R_\ell(a_j pe_i)$ and $\Psi_{max}(a_j pe_i) = \Delta_{max} + L_r(a_j pe_i) - R_\ell(a_j pe_i)$ as provided in equation 3.2.

Now we are ready to provide an algorithm for finding the maximal valid clusters for insertions as follows : traverse the genome from left to right and consider each locus g_i as a potential locus of insertion (i.e. the locus satisfying the first condition). For each of locus g_i take these steps:

1. construct the set of paired-end read mappings which can support an insertion at position g_i. This is the set
$$P_{g_i} = \{ape | (L_r(ape) < g_i < R_\ell(ape))\}$$

2. calculate the length of insertions supported by each paired-end read mapping in P_{g_i}. For each paired-end $ape \in P_{g_i}$ the length of insertions supported by the mapping is interval $[\Psi_{min}(ape), \Psi_{max}(ape)]$ (which can be calculated using equation 3.2). Thus the set $I_{g_i} = \{[\Psi_{min}(ape), \Psi_{max}(ape)] | ape \in P_{g_i}\}$ is the set of (intervals of) insertion lengths supported by each paired-end read mapping in P_{g_i}.

3. find all the maximal intersecting intervals on set of intervals I_{g_i} using $MInIn$ algorithm. The output of $MInIn$ is all the maximal valid clusters of insertion with assumption that locus of insertion is g_i.

Finally, after the algorithm reaches to the end of the reference genome, we need to discard valid clusters outputted by $MInIn$ (in the third step) which are proper subset of another cluster outputted by $MInIn$. Note that for each valid cluster created by $MInIn$ for locus g_i we only need to verify maximality with other clusters created by $MInIn$ for only loci in the range of $g_i - \Delta_{min}$ to $g_i + \Delta_{min}$.

Deletions

Deletion events have two breakpoints (denoted as Br_ℓ and Br_r) and finding both of them is the ultimate goal, however the paired-end read mapping strategy can not provide us the exact such breakpoints but a set of possible breakpoints. Thus for simplicity we have summarized all of such

possible breakpoints for a deletion into event $SV_{Del}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$. The event $SV_{Del}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$ represents a deletion located between loci Loc_L and Loc_R of the reference genome, with the length of deletion being in the range of Ran_{min} to Ran_{max}.

More formally given a valid cluster of paired-end read mappings $VClu$ there exists an event

$$SV_{Del}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$$

which includes all the possible deletion breakpoints supported by cluster $VClu$. We propose a similar algorithm for finding maximal valid cluster for deletions to the one proposed for insertions with some modifications. We use the formulation from [12] for finding the two potential breakpoints of deletions for given discordant mappings.

For each paired-end read mapping $a_j pe_i$ supporting a deletion (i.e. $R_\ell(a_j pe_i) - L_r(a_j pe_i) > \Delta_{max}$), the two breakpoints of the deletion should obey the equation 3.3. The potential left and right side breakpoints of the deletion are denoted by Br_ℓ and Br_r respectively.

$$\Delta_{min} < R_\ell(a_j pe_i) - Br_r + Br_\ell - L_r(a_j pe_i) < \Delta_{max} \tag{3.3}$$

The algorithm for finding all the maximal valid clusters for deletions works as follow: traverse the genome from left to right and consider each locus g_i as a potential left breakpoint of the deletion (i.e. assume $Br_\ell = g_i$). For each locus g_i take these steps:

1. construct the set of paired-end read mappings which can support a deletion with left breakpoint being locus g_i. Lets denote this set as P_{g_i}

$$P_{g_i} = \{ape | g_i - \Delta_{max} < L_r(ape) < g_i\}$$

2. calculate the right side breakpoints of potential deletions supported by each of the paired-end read mappings in P_{g_i} assuming the left side breakpoint of the deletion is g_i (i.e. $Br_\ell = g_i$). This can be done easily using the inequality 3.3. Note that for each of the paired-end read mappings in the set P_{g_i} the right side breakpoint would be an interval of possible loci in reference genome. Assuming that ape is a mapping in set P_{g_i}, then the right breakpoint of the deletion supported by ape (assuming that the left breakpoint is g_i) is an interval of loci denoted by $Br_r(ape, g_i) = [Br_{r_1}(ape, g_i), Br_{r_2}(ape, g_i)]$ where $Br_{r_1}(ape, g_i) = g_i + R_\ell(ape) - L_r(ape) - \Delta_{max}$ and $Br_{r_2}(ape, g_i) = g_i + R_\ell(ape) - L_r(ape) - \Delta_{min}$ (via equation 3.3).

WLOG assume $P_{g_i} = \{ape_1, ape_2, \cdots, ape_k\}$, then intervals of right breakpoints supported by each of these mappings is $I_{g_i} = \{Br_r(ape_1, g_i), Br_r(ape_2, g_i), \cdots, Br_r(ape_k, g_i)\}$.

3. find all of the maximal intersecting intervals of set I_{g_i} (using the optimal method $MInIn$). The maximal intersecting intervals of set I_{g_i} are potential maximal valid clusters for deletions with assumption that the left breakpoint of the deletion is locus g_i.

Finally, after the algorithm reaches to the end of the reference genome, we need to discard clusters created which are proper subset of another cluster created. Note that for each valid cluster created by $MInIn$ for locus g_i we only need to verify maximality with other clusters created by $MInIn$ for only loci in the range of $g_i - \Delta_{max}$ to $g_i + \Delta_{max}$.

Inversion

We represent a structural variation of type inversion by event $SV_{Inv}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$. The event $SV_{Inv}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$ represents an inversion located between loci Loc_L and Loc_R of the reference genome, and the length of inversion being in the range of Ran_{min} and Ran_{max}. Similar to deletion events, inversion events also have two breakpoints and the goal is to find these two breakpoints, however the paired-end read mapping strategy can not give us the exact such breakpoints, but a set of possible breakpoints. Thus for simplicity we represent all of such breakpoints for an inversion event by $SV_{Inv}(Loc_L, Loc_R, Ran_{min}, Ran_{max})$.

We propose a similar algorithm for finding maximal valid cluster for inversion to the one for deletion [4]. There are two differences between the algorithm for deletion and inversions and we will mention them here. First difference is the set P_{g_i}, which for inversions would be a more complicated than deletions. For inversions the set P_{g_i}, which denotes the set of all paired-end read mappings that can support an inversion when the left breakpoint of the inversion is g_i:

$$\begin{aligned} P_{g_i} = \{ape \quad | \quad & (or(ape) = ++ \wedge g_i - \Delta_{max} < L_r(ape) < g_i) \\ \vee \quad & (or(ape) = -- \wedge g_i < L_r(ape) < g_i + \Delta_{max})\} \end{aligned} \tag{3.4}$$

Second difference is the rules for calculating the potential breakpoints of inversions for each mapping in comparison to deletions. The rules for calculating potential inversion breakpoints for a

[4] We have modified the method proposed in [55] for producing maximal valid clusters. Similar to [12, 137] our modified method tries to match the two breakpoints. It should be noted that the method in [55] in some special cases could produced clusters which were not valid clusters for inversions (courtesy of Dr. Benjamin Raphael).

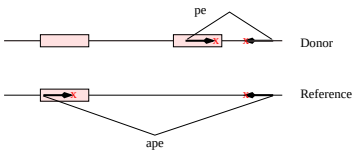
paired-end read mapping $a_j pe_i$ is given in equations 3.5.

$$\begin{aligned} or(a_j pe_i) = ++ \quad &\Longrightarrow \quad \Delta_{min} < Br_\ell - L_r(a_j pe_i) + Br_r - R_r(a_j pe_i) < \Delta_{max} \\ or(a_j pe_i) = -- \quad &\Longrightarrow \quad \Delta_{min} < L_\ell(a_j pe_i) - Br_\ell + R_\ell(a_j pe_i) - Br_r < \Delta_{max} \end{aligned} \tag{3.5}$$

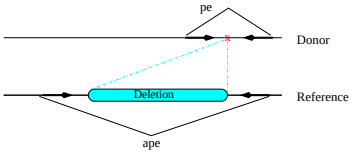
Transposition (mobile element insertions)

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*. In the remainder of this subsection we call such types of structural variants as *copy events*. Examples of common copy events include transpositions of Alu, SVA and L1 elements.

Unfortunately, none of the available methods designed to detect structural variation events (e.g. [144, 79, 73, 12, 84, 85, 55, 27, 137]) considered these copy 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 [118]. Interestingly, even if the goal of a method is to identify only insertions, deletions and inversions in a sequenced genome, the presence of the common copy events will cause many false negative deletion and inversion predictions. Figure 3.2 clearly demonstrates a common scenario where a copy event is mistakenly identified as a large deletion.



(a) The presence of a copy event 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 3.2: Transposon insertion causing 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 what follows, we study two classes of copy events and present the set of conditions based on the map locations and orientations of the paired-end alignments that imply a copy event within each

of these classes. First, we consider those copy 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 $SV_{\overline{Copy}}(Loc_L, Loc_R, Loc_{Br_L}, Loc_{Br_R})$.

For each copy event in reality there exists three breakpoints, which we will denote as Pos_L, Pos_R and Pos_{Br}. Pos_L represents the left locus of the segment being copied, Pos_R represents the right locus of the segment being copied and Pos_{Br} represent the locus (breakpoint) where the segment is copied to.

One of the following four cases should hold for a paired-end read alignment ape that supports a copy 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 3.3(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 3.3(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 3.3(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 3.3(d))

Similarly, one of the following cases should hold for a copy 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 3.4(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 3.4(b))

Case 3 ($Pos_{Br} > Pos_R$ and $or(ape) = ++$) : $\Delta_{\min} < Pos_{Br} - R_r(ape) + Pos_R - L_r(ape) < \Delta_{\max}$
(Figure 3.4(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 3.4(d))

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

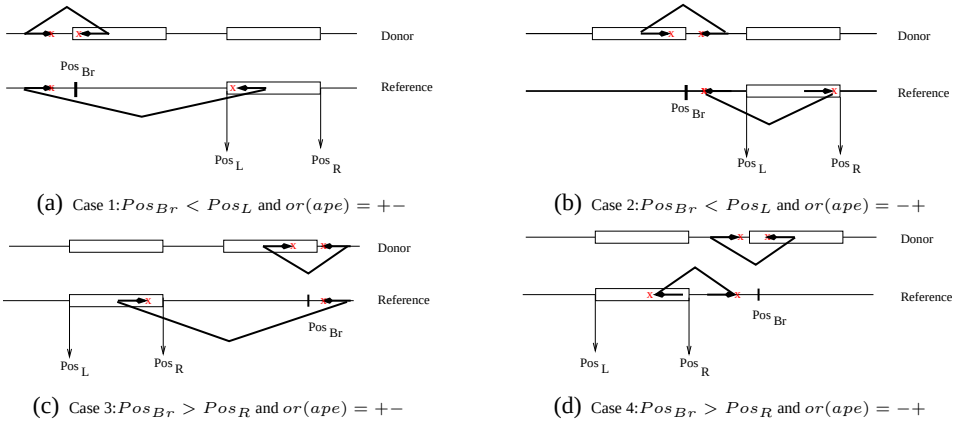
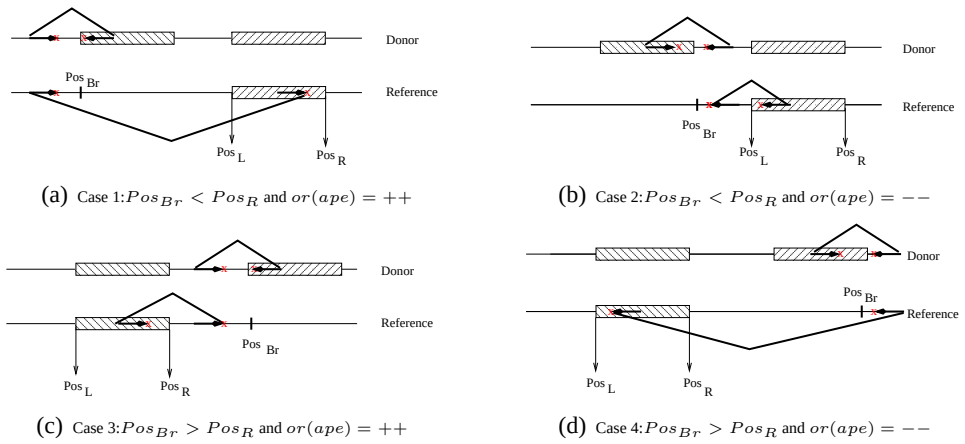


Figure 3.3: The set of conditions for each case that suggests a copy event in which the transposed segment is copied in direct orientation (Class I).

To find all maximal valid clusters for copy events, a naïve method would investigate all $O(n^3)$ possibilities of potential copy 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 for each possible values of Pos_{Br}, Pos_L, Pos_R, and then adding those paired-end reads that satisfy the conditions given in Section 3.1.4 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 in Section 3.1.4.



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

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 paired-end read mappings Ape^i. Thus, we are interested in finding *all maximal intersections* of the breakpoint intervals Br^i.

The algorithm $MInIn$ can output all the maximal intersecting intervals (of Br^i) in time $O(m_i \log m_i + s)$, where s is the size of the output. Please note that an algorithm which finds maximal valid clusters supporting copy of a given segment ϕ_i with running time of $O(m_i \log m_i + s)$ will yield an $O(m \log m + s')$ algorithm for finding maximal valid clusters for all the segments in Φ (since $m = \sum_{i=1}^{t} m_i$).

3.2 Novel Insertion Discovery

It is estimated that 19-40 Mb of human genomic sequence is missing from the human genome reference assembly [89]. Although high-throughput sequencing technologies have 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 a more 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 [73], and sequencing the entire fosmid clones [75]. 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" [89]). 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 [101]. 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 [147, 144, 79, 73] one cannot discover insertions > 100 bp when 200 bp insert size is used with the Illumina platform [18, 55, 85, 27]. Currently, the only method applicable for the discovery of long novel insertions using NGS technologies is *de novo* sequence assembly [136, 26, 90]. However, this approach requires large computational resources, and requires further processing to anchor the sequences to the reference genome.

Here we present a computational framework to discover the locus and content of novel sequence insertions using the NGS platforms. 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. For algorithms specifically designed for repeat sequence (or more formally, transposon) insertion detection as covered in section 3.1.

In Section 3.2.1, we will present the NovelSeq pipeline for novel insertion discovery, which consists of five different phases. In Section 3.4.4, we will discuss the results of the NovelSeq pipeline.

3.2.1 *NovelSeq* pipeline

NovelSeq is a pipeline to discover novel insertions into the donor genome in comparison to the reference genome. It has five steps, which is depicted in figure 3.5. Each of these steps is explained in detail in this section.

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 [4] and identifying *orphan* reads and *one end anchored (OEA)* reads. The paired-end reads where neither end-read [5] sequences can be mapped (with more than 95% similarity) to the reference genome are classified as *orphan* reads. Following the nomenclature previously described [73], if only one end-read is mapped onto the reference genome, such paired end reads are classified as One-End-Anchored (OEA). The set of One End Anchored reads is represented as OEA and the set of orphan reads is represented as $Orph$. The paired-end reads in OEA may also be mapped to multiple locations on the reference genome. Given $pe \in OEA$, $ape = ((Loc_\ell(ape), Loc_r(ape)), or(ape))$, where $(Loc_\ell(ape), Loc_r(ape))$ 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). 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 3.5).

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

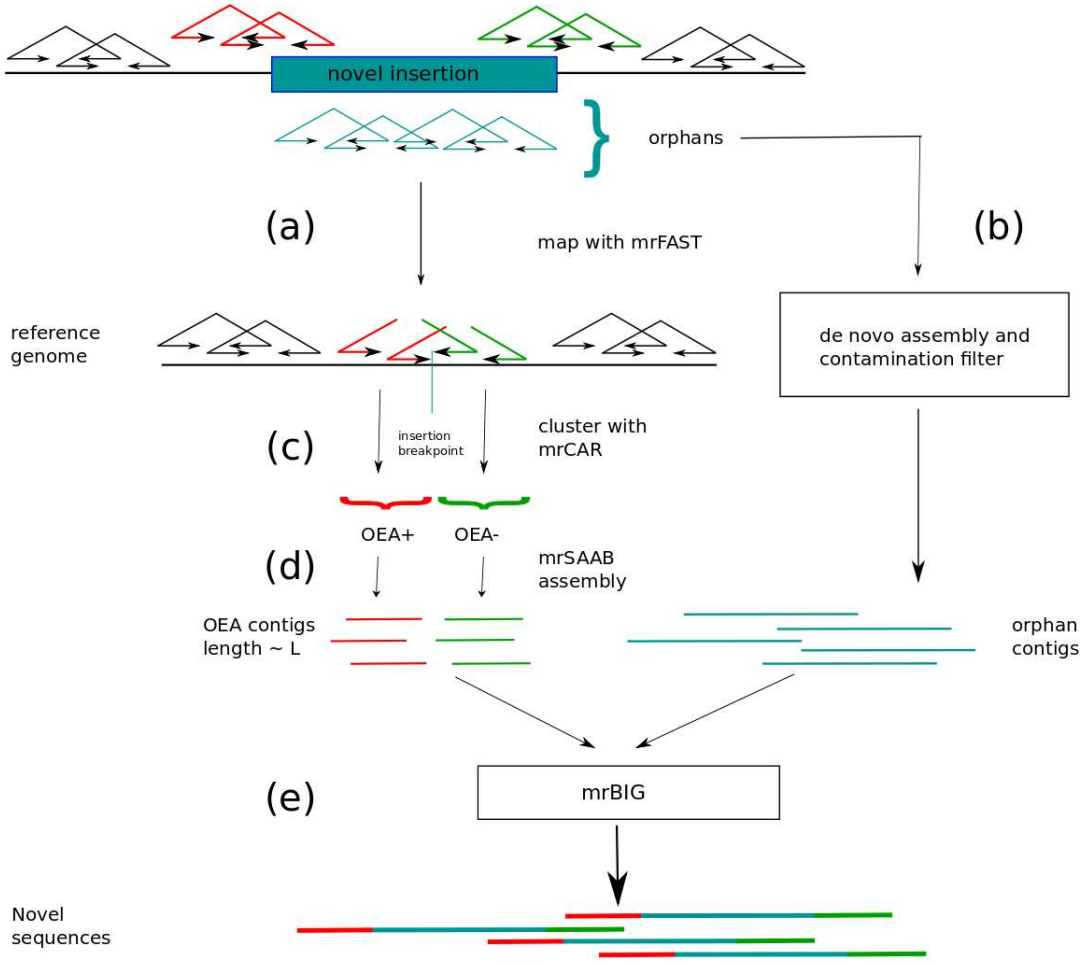


Figure 3.5: In this figure we illustrate the 5 stages of the NovelSeq pipeline. (a) Starts by mapping the paired-end reads to the reference genome, and classifies the paired-end reads to OEA and Orphan reads. (b) Assembles the orphan paired-end reads using available de novo assemblers, and removes any 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. Figure is taken from [47].

Orphan assembly and contamination removal

Using available de novo assembly algorithms such as EULER-SR [26] or ABySS [136], 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 [6], 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%.

Clustering the OEA reads and Selecting potential insertion locus

We formally describe a greedy algorithm to identify the OEA clusters and select the potential insertion loci (please see phase (c) in Figure 3.5).

A set of OEA reads $Clu \subset OEA$ supports the same insertion (i.e. are a valid cluster supporting novel insertion) if the following conditions hold:

- For every pair of OEA read mappings $\rho_F \in Clu$ and $\rho_R \in Clu$ where $or(\rho_F) \neq or(\rho_R)$ (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 $InsSize$ (i.e. $\Delta_{\max}$).
- The difference between the mapping locations of two OEA reads with different mapping orientations should not exceed twice the maximum $InsSize$ (i.e. $2\Delta_{\max}$).

Note that similar to 3.1.4 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. 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 g_i, we consider a window of size $2\Delta_{\max}+1$ centered at g_i. 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 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.

Similar to 3.1.2 we use the maximum parsimony objective function for selecting the insertions. 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. to minimize the total number of insertions) such that all OEA reads are aligned to the reference genome. We model this problem as a set cover problem and provide an $O(\log n)$ approximation solution. Note that the set of all OEA reads is the universe of elements, and the clusters created in the previous step are the sets that are selected to cover this universe. This is a necessary step since an OEA read can be present in multiple clusters. This is exactly similar to what we proposed in section 3.1.3 for other types of SV discovery using maximum parsimony assumption.

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, mrSAAB (micro-read Strand-Aware Assembly Builder). For each OEA cluster selected in previous phase (e.g. oea_i), the goal is to assemble the unmapped end of reads in oea_i with orientation $+$ into a single contig (i.e. oea_i+ contig) and the unmapped end of reads in oea_i with orientation $-$ into a single contig (i.e. oea_i- contig). This 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 (please see the phase (d) in figure 3.5).

The available *de novo* assemblers including EULER and ABySS do not provide the option of assembling the reads of only a single strand [6]. 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

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

follows, we briefly present this routine.

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 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.

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_F and oea_R 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 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., $oea_i = (oea_{i_+}, oea_{i_-})$). 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_+} and the score of alignment of prefix of or_i with oea_{j_-} 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 first $\Delta_{\max}$ base-pair of or_i and oea_{j_+} and the overlap between the last $\Delta_{\max}$ base-pair of or_i with oea_{j_-}.

Figure 3.6 shows how we reduce the merging problem into a bipartite matching problem.

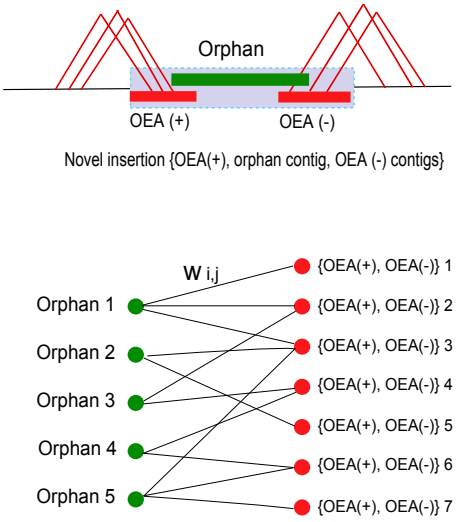


Figure 3.6: This figure illustrates how to reduce the problem of merging the orphan contigs with OEA contigs (note that each OEA cluster is in fact two contigs with different orientations, which together represent an insertion) into a maximum weighted matching problem in bipartite graphs. Each orphan 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. Figure is from [47].

3.3 Maximum Parsimony Structural Variation with Conflict Resolution

As mentioned before, the possibility of multiple map locations for each paired-end read raises the question of resolving which structural variants implied by the maximal valid clusters are *correct*.

The Maximum Parsimony Structural Variation (MPSV) problem as defined in section 3.1.2 aims to compute a unique mapping for each discordant paired-end read in the reference genome such that the total number of implied SV is minimized (in this section we consider all classes of SV). MPSV

was modeled as a combinatorial optimization problem and shown to be NP-complete. An approximation algorithm (denoted as VariationHunter) based on the *set-cover* problem with $O(\log n)$ approximation factor was given. However, the modeling of the MPSV problem imposed no limits on the number of "overlapping" SV predictions. A considerable amount of the predicated calls overlap with each other and a post-processing heuristic to filter some of those overlapping predicted SVs should be used (see the Results section 3.4.3). In this section we mathematically formulate these "conflicts" and model the structural variation discovery problem as a *novel* combinatorial optimization problem.

3.3.1 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 detection algorithm (such as VariationHunter, MoDIL or BreakDancer) might construct two or more sets of valid clusters that imply multiple conflicting deletion calls (Figure 3.7). Assuming the genome is haploid, it is not possible that both valid clusters in Figure 3.7 can be "correct".

We first formalize the Maximum Parsimony Structural Variation with Conflict Resolution (MPSV-CR) for both haploid and diploid genomes, and then we analyze the complexity of the MPSV-CR problem. Finally, we provide a heuristic solution for MPSV-CR. We call this solution as VariationHunter with Conflict Resolution (VariationHunter-CR).

Assuming a haploid genome, 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 variants implied by the two valid clusters, such that the existence of one of the events makes the other valid cluster incoherent (Figure 3.7). In the Appendix (section A), we present the set of rules to determine whether two valid clusters are in conflict in a haploid genome. Note that the rules in Appendix A are provided such that to consider any two valid clusters in conflict if there exists a configuration of the two potential SV's which makes one or both of the clusters not possible.

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

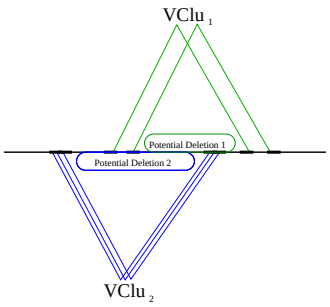


Figure 3.7: Two valid clusters $VClu_1$ and $VClu_2$ are shown to be in conflict in a haploid genome.

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, provided that the genome is diploid, both $VClu_1$ and $VClu_2$ might imply a correct SV in different haplotypes. Now consider a third SV cluster, $VClu_3$ that is in conflict with both $VClu_1$ and $VClu_2$. It is trivial that, based on the pigeon hole principle, $VClu_1$, $VClu_2$, and $VClu_3$ cannot simultaneously occur in a diploid genome. In other words, the presence of three different SV clusters that are conflicting pairwise in a haploid genome [7] will be a conflict in a diploid genome. The concept of the conflict graphs for haploid genomes can be replaced with *conflict hypergraphs* for diploid genomes. In the conflict hypergraph each hyperedge connects three nodes, if these three nodes are in conflict with each other [8].

Now, we define the Maximum Parsimony Structural Variation with Conflict Resolution (MPSV-CR) problem that does not only aim to minimize the total number of implied structural variants but also guarantees that no pairwise conflicting triplet of the implied SVs exists.

Note that this new version of the MPSV problem does not select any conflicting structural variants and thus may not always be able to assign every paired-end read to a particular SV. Thus the optimization function and constraints for MPSV-CR should be changed. One possible way is to allow some paired-end reads not to be assigned to a structural variation, however they should be penalized. Thus the new MPSV-CR problem not only tries to minimize the number of SVs predicted, but also maximizes the number of paired-end reads that can be mapped to reference 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.

[7] According to our definition of conflicts in a haploid genome.

[8] It should be noted that in the most general definition of conflict hypergraph, every odd number of nodes which creates a cycle in conflict graph should be connected by a conflict hyperedge (i.e., they are in conflict in the diploid genome).

3.3.2 Formal definition of the MPSV-CR problem

In this section, we formally define the MPSV-CR problem. Let $MC = \{VClu_1, \cdots, VClu_n\}$ be the set of SV clusters and $R = \{pe_1, pe_2, \cdots, pe_m\}$ be the collection of discordant paired-end reads. These discordant read pairs can be mapped to multiple locations in the genome, represented by $Align(pe_i) = \{a_1 pe_i, a_2 pe_i, \cdots, a_j pe_i\}$.

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 follows. Between every *three* distinct SV clusters that are pairwise in conflict, there exists a hyperedge in $E(CG)$:

$$\begin{aligned} E(CG) \quad = \quad & \{(VClu_i, VClu_j, VClu_k) \mid VClu_i, VClu_j, \\ & VClu_k \text{ are pairwise in conflict}\} \end{aligned}$$

Note that for the case when we only deal with a haploid genome (rather than a diploid genome), the hypergraph is nothing else than a simple graph (denoted by G) where each $e \in E(G)$ represents a pair of conflicting SV cluster.

We define a subset $SC \subset MC$ is *satisfiable* under the constraint hypergraph CG, if

$$\nexists e = (VClu_p, VClu_q, VClu_r) \in E(CG) : e \subseteq SC$$

(in the case of a haploid genome a subset $SC \subset MC$ *satisfiable* under the constraint graph G, if $\nexists e = (VClu_p, VClu_q) \in E(G) : e \subseteq SC$).

For each satisfiable subset SC and each paired-end read pe_i, we define the indicator variable $\Delta(SC, pe_i)$ as follows:

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

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 and the number of paired-end reads that are not assigned to any SV clusters.).

3.3.3 Computational complexity of MPSV-CR

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.).

Proof. 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 $k' = k/l$.

We prove that the MPSV-CR problem is NP-hard by using a reduction from the minimum set cover problem. Given C, a collection of subsets of a finite set S ($|S| = n$), we would like to find a subset $C' \subseteq C$ with the minimum cardinality such that every element in S belongs to at least one member of C'. Without loss of generality, we can assume given an instance of the set cover problem, we build an instance of MPSV-CR as follows:

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 select a satisfiable set of clusters SC such that $g(SC) \leq t$. On the other hand, if we can select a satisfiable set of clusters SC such that $g(SC) \leq t$ which includes x clusters and y *uncovered* 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 $pe_i, pe_{i+n}, \ldots, pe_{i+n(p-1)}$ and corresponding to each set $C_j \in C$ there is a cluster $VClu_j = \{pe_k | S_{k \bmod n} \in C_j\}$. We define $R = S$, $V(CG) = C$, and $E(CG) = \emptyset$ like in Case 1. If we have a set cover C' of size $\leq t$, we can select a set of clusters SC such that $g(SC) = kt$. When we can select 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. □

In the following, we show an inapproximability result even when we deal with a haploid genome.

Theorem 2. *There is no constant $\epsilon > 0$ for which MPSV-CR problem when $l = 1$ and $k = 1$ on haploid genome can be approximated within a factor of $n^{1-\epsilon}$ in polynomial time, unless $P = NP$.*

Proof. We use an approximation preserving reduction [112] from the Minimum Independent Dominating Set (MIDS) problem. Given a graph $G = (V, E)$ where $|V| = n$, the MIDS problem asks for a set $S \subset V$ with the minimum cardinality such that S is not only a dominating set but also an independent set of G. S is a dominating set of G if for each $v \in V$, either $v \in S$ or v is adjacent to some $v' \in S$. S is an independent set of the graph G if $\forall e \in E$: $e \nsubseteq S$.

Given an instance of the MIDS problem (denoted by Γ), we build an instance of MPSV-CR problem (denoted by Λ) as following: Corresponding to each vertex v_i in MIDS instance Γ, we have a cluster $VClu_i$ and a read r_i in MPSV-CR instance Λ. We also set $MC = R = V(G)$, $V(CG) = V(G)$, and $E(CG) = E(G)$. Now for each $i \leq n$, we define the SV cluster $VClu_i = \{v | \exists e \in E : e = v_i v\} \cup \{v_i\}$ (i.e. $VClu_i$ consists of vertices that are adjacent to v_i and includes v_i). Here we can give three properties of such a reduction:

1. It is easy to see that any independent dominating set to graph $G(V, E)$ with size K (i.e. lets assume set $\{v_i, v_i', \cdots, v_i^{(K)}\}$) has an equivalent set of K clusters in MPSV-CR instance which cover all the reads and there is no edge between the K clusters (i.e. the set of clusters $\{VClu_i, VClu_i', \cdots, VClu_i^{(k)}\}$) in constraint graph CG.

2. For any set of clusters SC which satisfy the constraints of MPSV-CR instance problem Λ and cover all the reads, there exists an equivalent set of nodes in graph $G(V, E)$ which has the same cardinality and is a independent dominating set for instance Γ.

3. Finally, for any set of clusters SC satisfying the constraints of MPSV-CR instance Λ (with cost $g(SC)$), we can find another set of clusters SC' satisfying the constraints of instance Λ where $g(SC') \leq g(SC)$ and all the reads in R are assigned to clusters in SC'. To construct SC' we use an iterative method and start by including all the cluster in SC to SC'. Lets assume read $r_i \in R$ is one of the uncovered reads by SC', then adding the respected cluster $VClu_i$ (which has r_i as a member) to SC' will still satisfies the conditions of MPSV-CR problem Λ. We know there is no edge between $VClu_i$ and any other clusters in SC', because r_i was not covered by any cluster in SC' and that means cluster $VClu_i$ or none of its neighbors in graph CG are members of SC'. Thus there is no edge between $VClu_i$ and any cluster in set SC'. It is obvious that such addition would not increase the cost (i.e. $g(SC' \cup VClu_i) \leq g(SC')$,

because the extra cost of adding a cluster will be canceled by profit of covering at least one more element). After several iterations of adding a clusters to SC' in each iteration, we can guaranty that SC' will cover all the reads and has a cost not greater than $g(SC)$.

It is not hard to see that these three properties show that the reduction provided is an approximation preserving reduction from minimum independent dominating problem to MPSV-CR problem. Hence, if MPSV-CR has an ϵ-approximation algorithm ($\epsilon > 1$) with polynomial running time, the MIDS problem also has a polynomial time approximation algorithm within the same factor. However the MIDS problem does not have any polynomial approximation algorithm within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$ unless $P = NP$ [49]. Thus, the MPSV-CR problem is not likely to have a polynomial approximate algorithm within the same factor.

□

3.3.4 An efficient solution to the MPSV-CR problem

In this section we present a heuristic solution for a special case of the MPSV-CR problem where both k and l (the coefficients in the optimization function g) are set as $k = 1, l = 1$. This heuristic, named as $Max\_Assigned\_Reads$, consists of two phases:

In the first phase, we form a *maximal satisfiable set* of SV clusters (denoted by MS) in a greedy fashion. Note that a satisfiable set of SV clusters, SC (as noted in the previous section), is called *maximal* if no other SV cluster (such as $VClu$) can be added to SC (i.e. $VClu$ together with two existing SV clusters in SC will form a hyperedge in the conflict hypergraph CG). We start with $MS = \emptyset$ and then iteratively add the SV cluster that covers the most number of paired-end reads [9] to MS until MS becomes a maximal satisfiable set. Note that we add an SV cluster SV_k to MS in an iteration even if SV_k does not cover any *new* discordant paired-end read (provided that MS remains satisfiable). We denote MR as the set of all paired-end reads covered by the SV clusters in MS.

In the second phase of $Max\_Assigned\_Reads$, the aim is to select the minimum number of SV clusters from MS that cover all paired-end reads in MR. For this phase we use a set cover approach similar to solution for original maximum parsimony structural variation discovery as mentioned in section 3.1.2.

In what follows, we give a lower bound on the cardinality of MR. The analysis of the second

[9] We count the paired-end reads that were not previously covered by any SVs in MS

phase of $Max\_Assigned\_Reads$ is similar to [55]. Let m be the total number of discordant paired-end reads, and let $neighbors(VClu) = \{VClu' | \exists e \in E(CG), VClu, VClu' \in e\}$, $deg(VClu) = |neighbors(VClu)|$ and $\Delta = \max\{deg(VClu) | VClu \in MC\}$ i.e. the maximum degree of a vertex in the conflict graph CG.

Theorem 3. $|MR| \geq m/(\Delta + 1)$.

Proof. Let k be number of iterations of $Max\_Assigned\_Reads$. For each i ($1 \leq i \leq k$), we denote $VClu_i$ as the cluster that is selected at the i^{th} iteration. We also denote MR_i as the set of paired-end reads that are covered by $VClu_i$ for the first time. Furthermore, we define UR_i as the set of paired-end reads that are not covered by any of the SV clusters $VClu_1$ through $VClu_i$ and also is not able to be covered later (as the result of selecting $VClu_i$) in the remaining $k - i$ iterations.

At the ith iteration, the maximum number of reads that could be covered for all neighbors of $VClu_i$ is at most $\Delta|MR_i|$, thus $|UR_i| \leq \Delta|MR_i|$. Moreover, we have $\sum_{i=1}^{k} (UR_i + MR_i) = m$ and $|MR| = \sum_{i=1}^{k} MR_i$. Hence $m/(\Delta + 1) \leq |MR|$. □

3.4 Experimental Results

3.4.1 Best Mapping vs All Mapping

We used the simulated paired-end reads produced in [122] to study the effect of using VariationHunter with multiple mapping in comparison to using best mapping for each paired-end read [10]. Simulated paired-end read data that approximates the fragments sizes and read lengths that are routinely obtained with short read, short insert sequencing technologies were used in this experiment. For details of this simulation (data production and alignment) please see [122] [11]. Since VariationHunter utilizes reads with non-unique alignments, discordant pairs that have multiple alignments to the reference were also considered. Considering the fact that GASV [137] produces maximal valid clusters (a.k.a maximal intersecting regions) as structural variations, thus the difference observed between VariationHunter and GASV is due to using multiple mapping with maximum parsimony by VariationHunter in contrast to best mapping in GASV. Figure 3.8 shows the comparison of VariationHunter and GASV in predicting simulated deletions. As it can be seen considering multiple mappings for structural variation discovery clearly gives advantage over best mapping approach.

[10] Courtesy of authors of [122] for providing us with the simulation data.

[11] It should be noted that the difference between VariationHunter results shown in figure 3.8 and the one shown in [122] is due to using different parameters for VariationHunter.

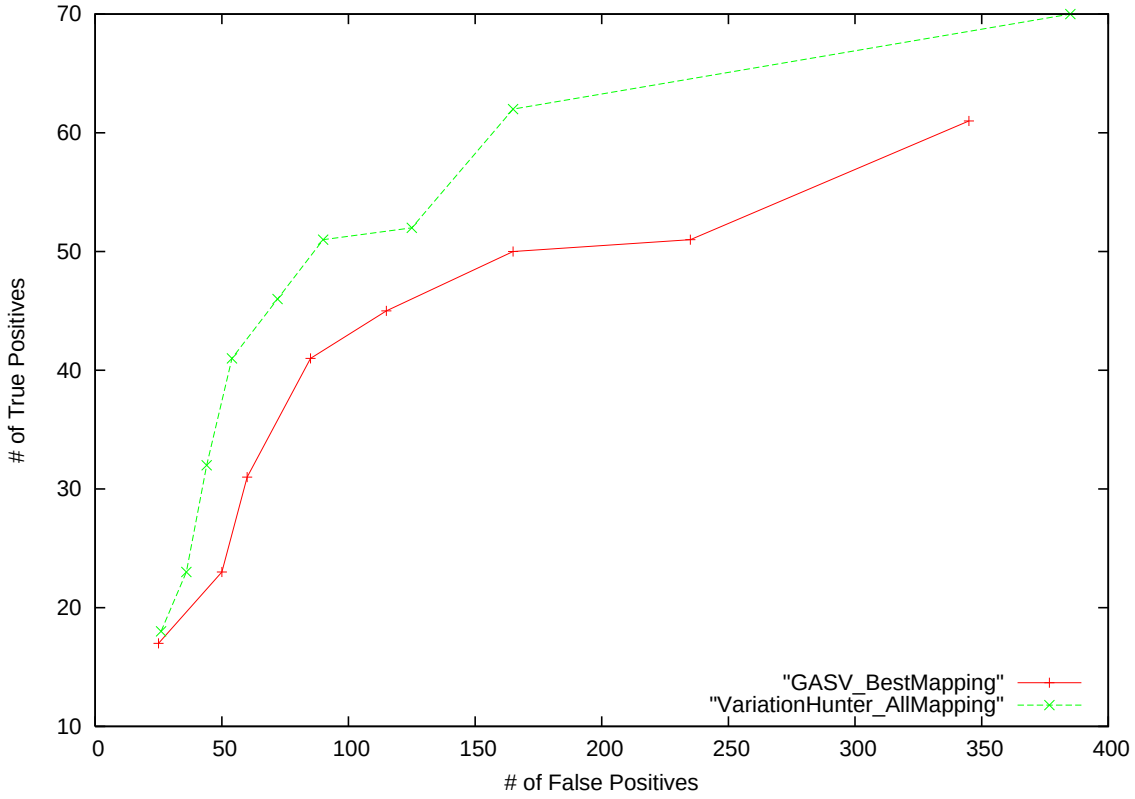


Figure 3.8: The comparison of unweighted VariationHunter and GASV in capturing simulated deletions. It is important to note that for GASV best mapping was used, while for VariationHunter multiple mappings were considered.

3.4.2 Mobile Element Insertion Discovery

Implementation of Mobile Element Insertions

It is established that there are a large number of mobile elements in the human genome [105]. For example, the reference human genome assembly annotation includes 1 million copies of the Alu element. Considering all known mobile elements (segments in genome) as potential transposon sequences for our algorithm would be very time consuming, and in fact, unnecessary. The consensus sequences for all mobile elements are well studied, and available at the RepBase database [70]. We used the consensus sequences of these mobile element families as representative sequences to facilitate faster experimentation. To this end, we create a new sequence, denoted as chrN: we first append a poly-N sequence to each consensus sequence, and then concatenate them, generating chrN. In these experiments we considered only Alu and SVA elements.

For read mapping we use mrsFAST [46], a cache-oblivious short read mapper recently developed as an extension of mrFAST [4]. mrsFAST maps the paired-end reads to *all* locations with Hamming distance less than a user-defined threshold ω. In this experiment we set $\omega = 2$.

Given paired-end whole-genome shotgun sequence library $R = \{pe_1, pe_2, \cdots pe_n\}$, we follow the following steps to obtain the reads (and mapping information) for transposon discovery:

- We first map all paired-end reads to chrN using mrsFAST and discard such paired-end reads that cannot be mapped to chrN. We keep the read pairs where one end-read is mapped to chrN.
- Next, we map the reads we determine in the previous step to the reference genome, and discard all paired end reads with at least one *concordant* mapping.
- Finally we re-map the reads from previous step to both chrN and the reference genome. As a post-processing step, we select the paired-end alignments where one end is mapped to chrN and the other end is mapped to the reference genome.

Mobile element insertion discovery in the Venter genome

A list of mobile element insertions in the Venter genome assembly (HuRef) [87] in comparison to reference human genome assembly (NCBI build 36) was recently published [154]. We used the available HuRef genome to produce short paired-end reads, similar to reads generated by the Illumina technology (simulating an Illumina sequencing of the Venter genome) to benchmark the sensitivity and specificity of our algorithms. To our knowledge this is the only dataset for mobile element insertion annotations from the genome of a single individual. We created paired-end reads from the HuRef genome with a read length of 36 bp, and obtained 10-fold sequence coverage. The fragment insert sizes for paired-end reads were chosen randomly that follows a normal distribution very similar to the fragment size distribution in the NA18507 shotgun sequence library generated using the Illumina platform [18].

We used our mobile element insertion discovery feature of VariationHunter to discover transposons in the autosomes of HuRef (from chr1 to chr22). In our experiments, we focused on Alu, NCAI (Non-classical Alu Insertion)[12], and SVA insertions. The results of our experiment are summarized in Table 3.1. The validated set of mobile element insertions in the HuRef assembly is a union of the published transposon insertions in [154] and our *new* transposon predictions not listed in [154] but are included in HuRef.

[12] Alu insertions that only contain the internal fragment of Alu are called Non-classical Alu insertions (NCAI) [154].

As shown in Table 3.1, our method was able to find most of the known/validated mobile element insertions (recall rate is $> 85\%$) while the number of invalidated predictions is very low (precision rate $\sim 90\%$).

Interestingly, most of the Alu insertions missed by our algorithm were truncated insertions (significantly smaller than the consensus sequences). Figure 3.9 summarizes the true positive / false negative results with respect to the length of Alu insertion.

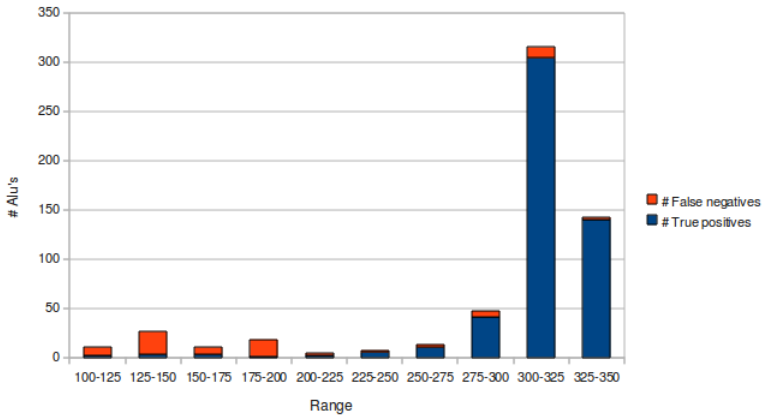


Figure 3.9: The length distribution of true positive and false negative mobile element insertion predictions for Alu elements. Note that all of the Alu consensus sequences used in creating chrN are longer than 250 bp.

3.4.3 Deletion and Inversion Discovery on NA18507

We tested VariationHunter with the paired-end read WGS library generated from the genome of an anonymous donor (NA18507) using the Illumina technology [18] for prediction of deletions and inversions on this genome. We first 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[13]. This constitutes $\sim 42X$ sequence and $\sim 120X$ physical coverage of the human genome. Using longer inserts (40Kbp fosmids), a set of structural variation events in the genome of the same individual was previously detected and experimentally validated [73], which we also utilize to test the sensitivity of our algorithms, and compare against the sensitivity of the method used in [18] (see table 3.2).

Due to the lower sequence quality generated by the Illumina platform, we first pre-screened the paired-end reads. We removed any mate-pairs from consideration if either (or both) end sequences

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

Table 3.1: Summary of mobile element insertion prediction results in the Venter genome. We show the precision and recall rates of our mobile element (Alu, NCAI, and SVA) insertion discovery. We compare our mobile element insertion predictions with both [154] and the HuRef genome assembly[87]. The results demonstrate that our algorithm has a high recall and precision rate.

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%

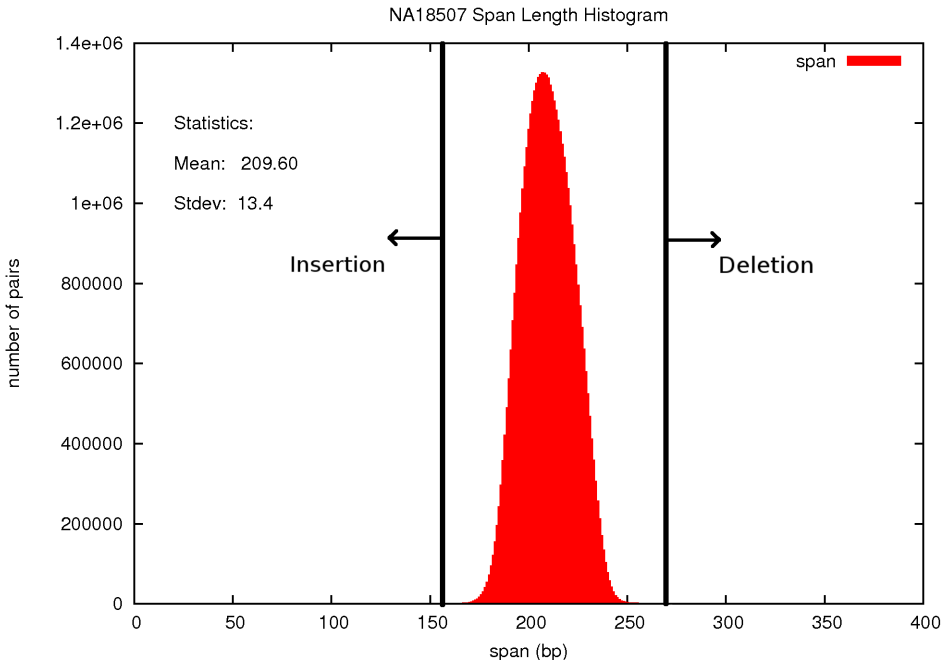


Figure 3.10: Span length histogram of paired-end reads from NA18507 on human genome build 36. We call a paired-end read concordant if its span is within 4 std of the mean length. For this library, the concordant size cut off values are 155bp and 266bp.

has average *phred* [38] quality value less than 20, or if either (or both) sequences contain more than two 'N' characters. This resulted in the removal of ~ 1.3 billion reads. Approximately 50 million pairs of sequences were first sampled and mapped to the reference genome to establish length statistics (average span length 209bp and standard deviation (std) 13.4bp - see Figure 3.10 for the length distribution histogram). All ~ 2.2 billion higher-quality end sequences were then mapped to the reference genome using our in-house sequence mapping tool *mrFAST* [4], and all possible locations within an edit distance of 2 were considered [14]. In total, 2.17 billion reads (95.9%) were mapped to the reference genome (~ 804 million inserts where both ends mapped) yielding approximately $56X$ physical coverage over the reference genome. We then discarded any mate-pairs where the total number of paired configurations exceed $1,000$, or if either of the end sequences map to more than $5,000$ locations. The read mapping stage was completed within a week using 200 CPU cores on our

[14]The reader should also note that our algorithm is compatible with any sequence mapping tool that can return multiple map locations, such as Mosaik [51] or SHRiMP [155]

	Validated [73]		VariationHunter						Bentley et al.		
			Weighted			Unweighted					
			Pred.	Capt.		Pred.	Capt.		Pred.	Capt.	
Validation type	S	L		S	L		S	L		S	L
Deletion	92	143	8,959	57	85	7,599	55	82	5,704	49	67
Inversion	13	82	504	2	23	433	4	25	*NA*	*NA*	*NA*

Table 3.2: Comparison of structural variation detected in the Illumina paired-end read library generated from the genome of NA18507 with the validated sites of variation (sample (S) and locus-level (L) validation; remapped to human genome build 36) using fosmid based approach from the same individual. We require that at least 50% of either the validated or predicted deletion interval to be covered to call an overlap. Inversions are considered to be captured if there is any intersection between the validated and predicted interval. The original study with the Illumina data does not report the inversion calls, primarily because inversions usually flanked by repeat sequences which were mostly missed by unique sequence mapping [18].

computational cluster. As per [144, 79, 73, 84] we call a clone insert *concordant* if it spans within $4 \times std$ of the average length (155bp to 266bp), and the mapping orientations of both ends obey the rules dictated by the WGS library [15]. In the end, we obtained $787,667,370$ concordant, and $16,766,282$ *discordant* (deletion, insertion, inversion, everted, translocation) pairs ($296,408,665$ discordant combinations). Concordant clones are removed from further consideration for SV detection, however, the concordant clone information can further be utilized to infer heterozygosity. VariationHunter algorithm was then run on the map locations of the discordant paired-end read sequences. It took less than 1 hour for the VariationHunter algorithm to complete on a single computer with AMD Opteron processor.

To increase our confidence in SV prediction by VariationHunter, we filtered variants with less than 5 supporting independent clones for the unweighted version, and variants with $weighted\_support < 3$ were filtered out in the weighted version ($weighted\_support$ is total summation of weights of paired-end reads supporting an SV. Note that weight of each paired-end supporting an SV is based on its sequence similarity score; it is also normalized such that total weights associated to each paired-end read is equal to 1). We also required at least 5bp between the start coordinates of map locations to eliminate possible duplicated reads [144, 73]. False positives frequently are recognized post-factor as too many clustered calls predicting more than 2 alleles. We applied a simple heuristic to remove the discrepancy between predicted events that would simply filter out the calls with

[15] For example in the Illumina platform (short insert library), the upstream end sequence is expected to map to the + strand, where the downstream end sequence is expected to map to the - strand.

less mate-pair support among the incompatible variants. In case of equal mate-pair support of 2 contradicting indels, we select the variant with smaller size. We further limited our deletion calls to $500Kb$ and inversion calls to $10Mb$ in size, and discarded any inversion events if only one of the two breakpoints are observed. Our first algorithm, *unweighted VariationHunter*, returned a total of $8,959$ deletions, 504 inversions, and $5,575$ insertions with the weighted version; and $7,599$ deletions, 433 inversions, and $3,772$ insertions with the unweighted version.

We compare the predicted deletions and inversions with both sample-level and locus-level validated sites of variation in Table 3.2. Structural variants detected by fosmid ESP approach that are validated using the fosmids from the same individual are classified as "sample-level validated". If a variant is predicted in multiple individuals (including NA18507), but validated with fosmids from another individual (to reduce cost and labor for validating common variants), then it is categorized as "locus-level validated". Our thresholds to call an overlap are more strict than the original Illumina study [18]. We require for deletions that the length of the intersection of validated and predicted overlap to at least 50% of the length of the union of the intervals; or the deletion variant predicted by Illumina sequencing to be entirely encompassed by the fosmid interval (due to the difference between detection resolutions of the two methods). Any overlap between the predicted and validated inversion calls are considered as captured. In total, we were able to predict $\sim 62\%$ of the validated deletions with our VariationHunter, where the original study shows evidence for $\sim 53\%$ overlap when same thresholds are applied. Our true positive rate can be improved to $64/92(70\%)$ for sample-level validated, and $96/143\ (67\%)$ for locus-level validated sites by simply removing the $weighted\_support \geq 3$ requirement, but this also increases the total number of deletion prediction to $13,320$ sites (134Mbps) as opposed to $8,959$ intervals (23.4Mbps). Note that, it is impossible to capture any insertion events larger than the difference between the average insert size and the concordant size cut off ($209 - 155 = 54$bp in this set) using paired-end reads without the use of sequence assembly methods. However, since the average length of a fosmid clone is 40Kbp, Kidd et al. [73] could detect insertions of size only between 800bp-8Kbp, thus insertion predictions from these two sets are not comparable.

Although the number of deletions we discover using the Illumina WGS set is significantly higher than the validated sites, 24.5% of the predicted intervals report less than 100bp, and 96.75% of the intervals call < 8Kbp of deletion (Figure 3.11), which is the resolution limit for deletion detection with fosmid ESP analysis [144, 73]. This is mainly due to the ability to detect shorter deletions with the smaller insert size (200bp against 40Kbp).

In addition, the length distribution of the predicted sites of deletion (Figure 3.11) also show

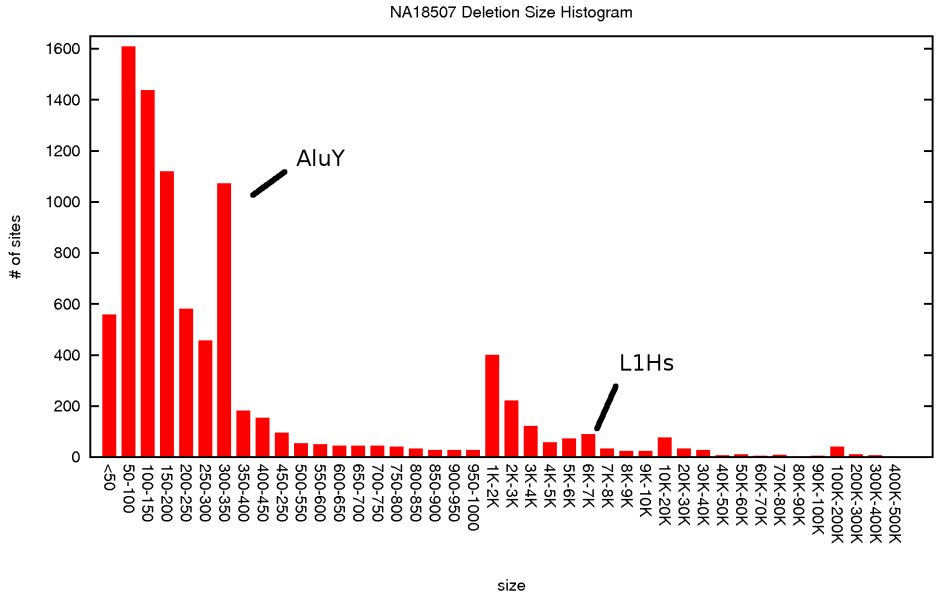


Figure 3.11: Deletion length histogram of detected SVs from NA18507 on human genome build 36 with weighted VariationHunter algorithm. Increased number of predicted deletions of size 300bp and 6Kbp (due to AluY and L1Hs repeat units respectively) are clearly seen in the histogram, confirming the known copy-number polymorphism in retrotransposons [15, 21].

increased number of 300bp and 6Kbp deletions confirming the known copy number polymorphism of retrotransposons [15, 21]. We also showed the length distribution of deletion sites for lengths $> 100bp$ called by our weighted VariationHunter (which is total of 6759 predicted deletions) and the set of homozygous deletions in comparison of individual (J. Craig Venter) and NCBI human reference assembly [87] (which is a total of 3104 deletions of length $> 100bp$) in Figure 3.12.

3.4.4 Novel Insertion Discovery on NA18507

We tested our NovelSeq 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 [18]. Similar to the pre-screening methodology used for other types of SV discovery 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

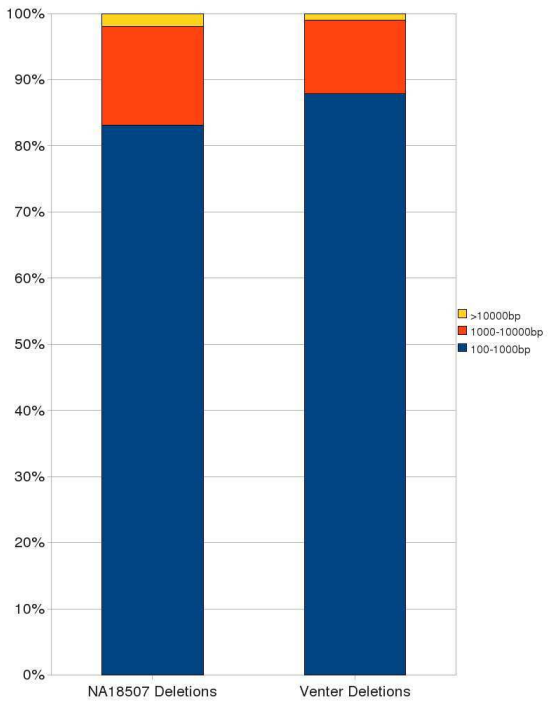


Figure 3.12: Comparison of deletion size distributions detected from the genome of NA18507 with the weighted VariationHunter algorithm and from Venter genome as reported in [87].

more than 2 unknown (i.e. N) characters. 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* [4], 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*. Using ABySS [136], we assembled the orphan paired-end reads into $4,154$ contigs of size $\geq$200bp. (N50 = 995). In the rest of this section we call these contigs as *ABySS contigs*. As an independent assessment, we also generated the sequence assembly of the orphans using the EULER [26] algorithm, which we call *EULER contigs*. EULER returned $4,564$ contigs of size $\geq$200bp. (N50 = 730). Next, we screened the orphan contigs to test for contamination. Using BLAST [6], we compared the orphan contigs with the database, 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.

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 ≥ 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.

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* [16] 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. Resulting OEA contigs were then processed together with the orphan contigs in the last phase of the NovelSeq pipeline. 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 orphan contigs can be anchored to the reference assembly.

Recently, Kidd et al. [73] sequenced all fosmid clones ($\sim$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 [73]. We used this dataset as the gold standard to test the accuracy of the NovelSeq pipeline. As shown in Table 3.3, 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 $\geq$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

[16] 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.

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 3.3: 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 $\geq$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.

3.4.5 Structural Variation Prediction with VariationHunter-CR

In this section we show that the call set 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 an experiment we compare the deletion predictions using different algorithms including VariationHunter-CR in the genome of a Yoruba African donor (NA18507) sequenced with the Illumina Genome Analyzer platform [18]. We include the validated deletions detected in the genome of the same individual using fosmid clone-end sequencing [73] in our comparisons.

Figure 3.13 shows the comparison of our new VariationHunter-CR algorithm with the original VariationHunter, the curated (post-processed to remove conflicting predictions) result published in [55] and BreakDancer [27].

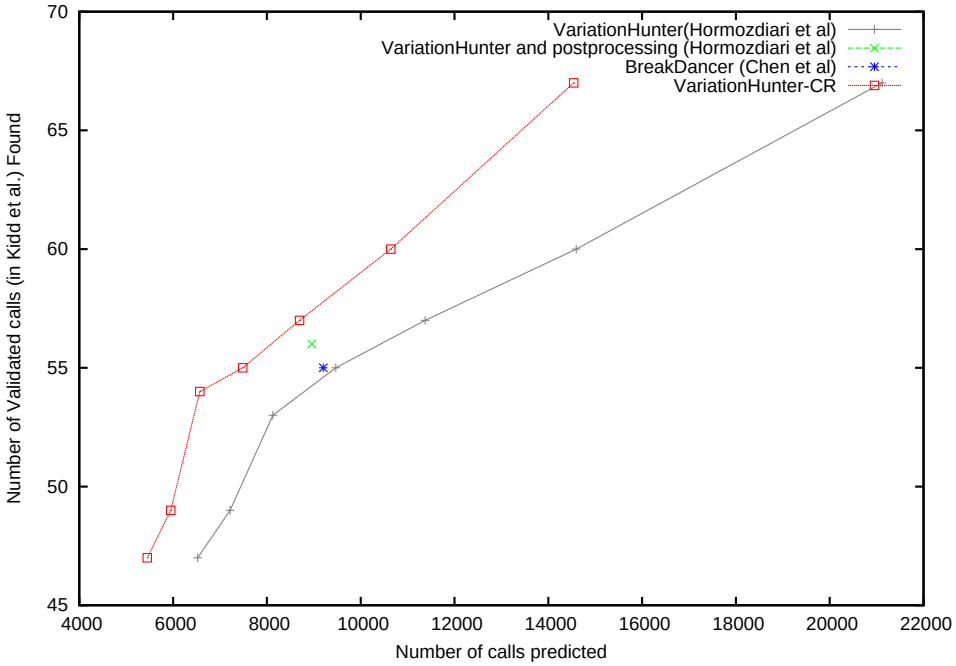


Figure 3.13: Prediction performance comparisons of VariationHunter (black), VariationHunter-CR (red) and BreakDancer (blue). We also show the curated (post-processed) results of VariationHunter in this figure (green). The x-axis represents the number of deletions predicted by each method, and y-axis is the number of validated deletions from [73] that overlaps ($> 50\%$ reciprocal overlap) with a prediction. It is desirable to obtain a prediction set is able to find more validated calls with less number of total calls 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); less support level results in more predicted deletion intervals. This figure shows that VariationHunter-CR has a better performance than VariationHunter for all support levels, and both VariationHunter and VariationHunter-CR outperform the BreakDancer algorithm [27].

Chapter 4

Alu transposition map on human genomes

The discovery of the Alu elements more than 30 years ago [131, 61] as 300 basepairs (bp) interspersed repeat sequences commonly found within the introns of genes [33] prompted an active area of research to address the role of mobile elements in genome evolution and human disease [13]. More than one million Alu retrotransposons comprise over 10% of the human genome sequence [82, 13, 64]. They are partitioned into numerous subfamilies, which have been active at different time points during primate evolution [117, 92]. Currently, 30 distinct categories of Alu subfamilies are recognized [105] with AluYa5 and AluYb8 being most active in the human lineage [24]. Alu retrotranspositions have numerous consequences leading to insertional mutations, gene conversion, recombination, alterations in gene expression, pseudogenization, structural variation and formation of segmental duplications [13, 8, 71, 154].

Traditional methods to detect Alu insertion polymorphisms involve polymerase chain reaction (PCR) where putative polymorphic loci are genotyped one by one [11, 127, 31]. Recently, PCR-based capture and high-throughput sequencing methods have been applied to quickly screen thousands of mobile element transposition events [37, 153]. 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 [73, 17, 74], transposon insertion profiling by microarray [62], and restriction enzyme profiling followed by Sanger and Roche 454 sequencing [68]. Whole-genome shotgun sequencing (WGS) of different individuals [87, 18, 150, 152, 98] 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 [154], and few studies have attempted to systematically discover these events at the individual genome level.

We described a computational method to discover mobile element insertions in genomes sequenced by paired-end next-generation sequencing (NGS) platforms in section 3.1 (also in [58]). Based on our structural variation detection algorithm, VariationHunter [55], our method follows the repeat anchored mapping approach [73, 95] to effectively cluster paired-end reads where one end maps to an annotated repeat element and its mate maps to a position within the genome. We previously demonstrated the sensitivity and specificity of our algorithm by simulation, proving its detection power (see section 3.4.2). Here, we 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.

VariationHunter discovers the mobile element insertions based on a maximum parsimony structural variation discovery algorithm [55]. In the first step, the algorithm clusters the discordant paired-end reads that support the insertion of an Alu element. Next, VariationHunter selects the minimum number of such clusters (mobile element insertions) that cover all paired-end reads [58]. For the YRI trio, we first pooled all discordant reads and then applied the VariationHunter algorithm on the combined set of read mappings. This pooling strategy takes advantage of a priori information that the variation between individuals within a trio should be limited.

4.1 Discovery and Validation

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 4.1). We considered individuals from different populations, including three Yoruban individuals from Ibadan, Nigeria (YRI: NA18506, NA18507, and NA18508 [18]), one Centre dÉtude du Polymorphisme Humain (CEPH) individual of European origin (Utah resident with ancestry from northwestern Europe, CEU: NA10851 [113]), two Khoisan individuals from southern Africa (KB1 [132] and HGDP01029 [44]), one Han Chinese (YH [150]), and one Altaic Korean (AK1 [76]). The three Yoruban genomes constitute a parent-child trio providing us the opportunity to study transmission of Alu insertions (Table 4.1). We computationally predicted novel Alu insertion loci using VariationHunter [58]. Briefly, we mapped the WGS data using mrFAST [4] to the reference genome

Individual	Population	# reads (M)illion	Read Length (bp)	Insert size (bp)	Min Support	# Alu	dbRIP	dbRIP + Others
NA18506	YRI	3444M	35	222	6	1720	294	440
NA18507 [18]	YRI	2261M	36-41	208	6	1579	292	435
NA18508	YRI	3175M	35	203	6	1744	310	451
NA10851 [113]	CEU	1309M	36-101	132-384	5	1282	370	501
AK1 [76]	Korean	1430M	36-106	132-384	2	909	225	327
YH [150]	Han Chinese	979M	35	135-440	3	1160	307	462
KB1 [132]	Khoisan	842M	36-37	181	2	457	92	144
HGDP01029 [44]	Khoisan	161M	76	150-300	2	307	60	93

Table 4.1: Genome of eight donors studied for Alu insertions.

(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 4.1).

In total, we predicted 2,451 novel Alu insertions not present in the human reference genome for the YRI trio sequence data (see Figure 4.1) and a total of 4,342 Alu insertions in the entire set (see Figure 4.2). We find that only 13.2% (571/4,342) of these loci have been previously reported in the database of retrotransposon insertion polymorphism (dbRIP) [149]. If we include two additional, recently published surveys [68, 153], we find that 79.0% (3,432/4,342) of our calls are novel (dbRIP+other column in table 4.1). 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 4.3). We identified 31 Alu elements that retrotransposed within an exon, of which three are predicted to disrupt a coding sequence (please see Figure 4.4). Also the list of Alu insertions falling inside exons and coding exons see Tables B.1, B.2 in Appendix B.

We experimentally validated a set of Alu insertion predictions from seven of the eight genomes using PCR. We selected 29 sites from the YRI trio where integrations had occurred in relatively unique genomic regions facilitating PCR primer design. All 29 sites and the transmission genotypes within the trio were validated by PCR. We then tested rare Alu insertions that were predicted to be specific to an individual targeting the genomes of NA10851, AK1, KB1, and YH. We performed PCR on 10 selected sites from each of these four genomes. We removed five of the 40 PCR assays from consideration due to amplification failure, and only 25 of the remaining 35 sites confirmed the

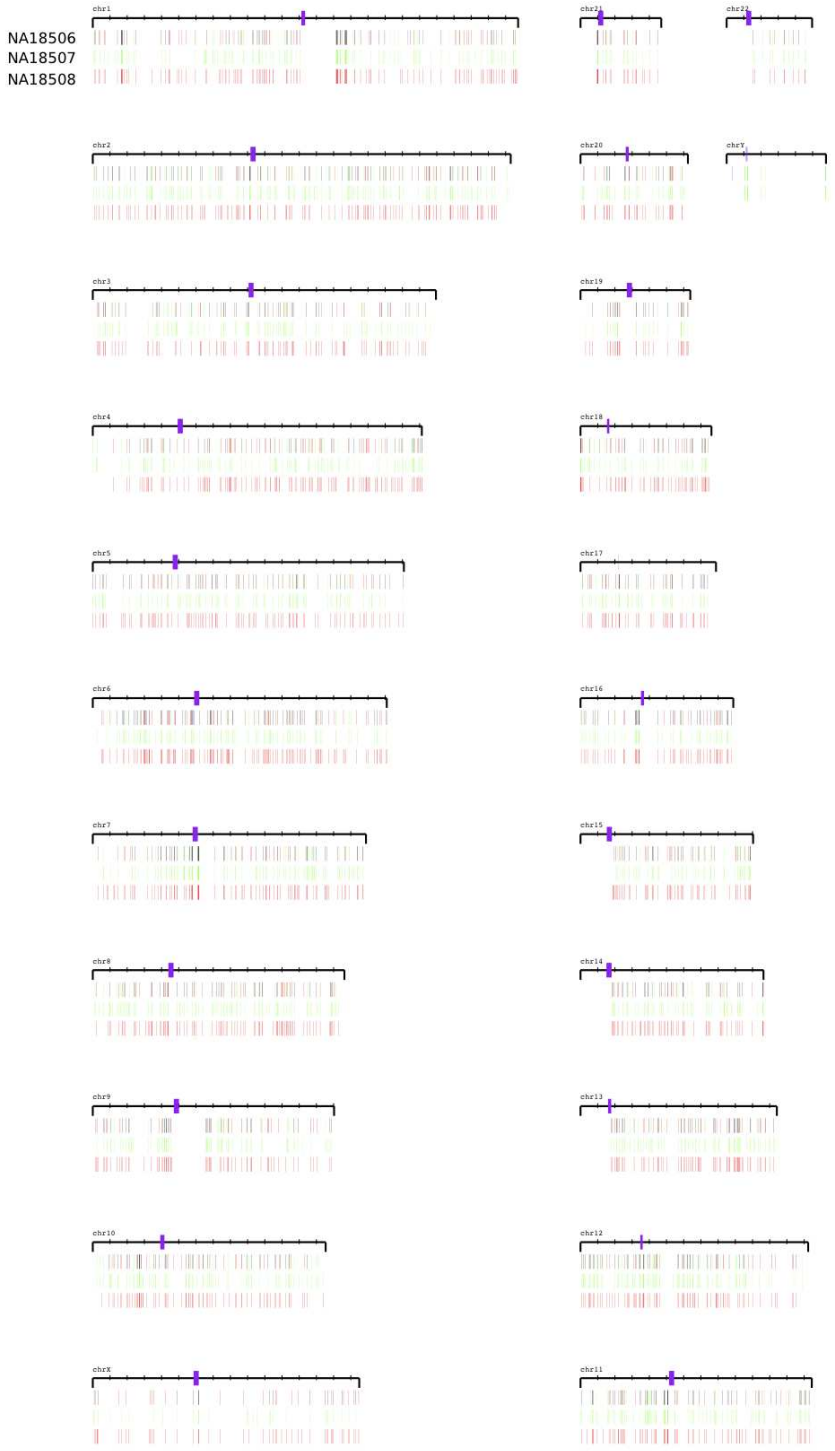


Figure 4.1: The Alu insertion loci are shown for the YRI trio in the order of NA18506, NA18507 and NA18508.

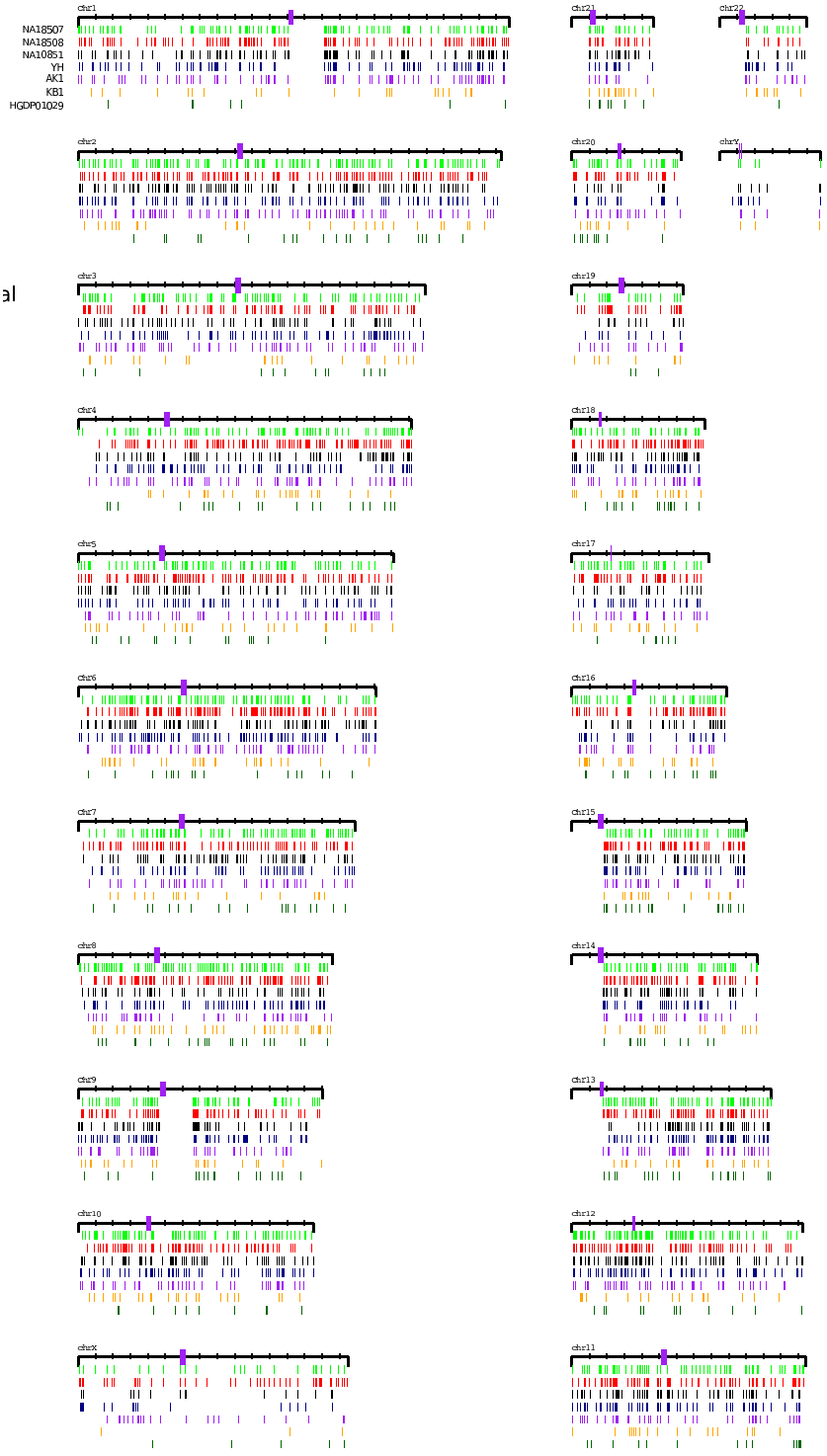


Figure 4.2: The Alu insertion loci are shown for 5 different individuals

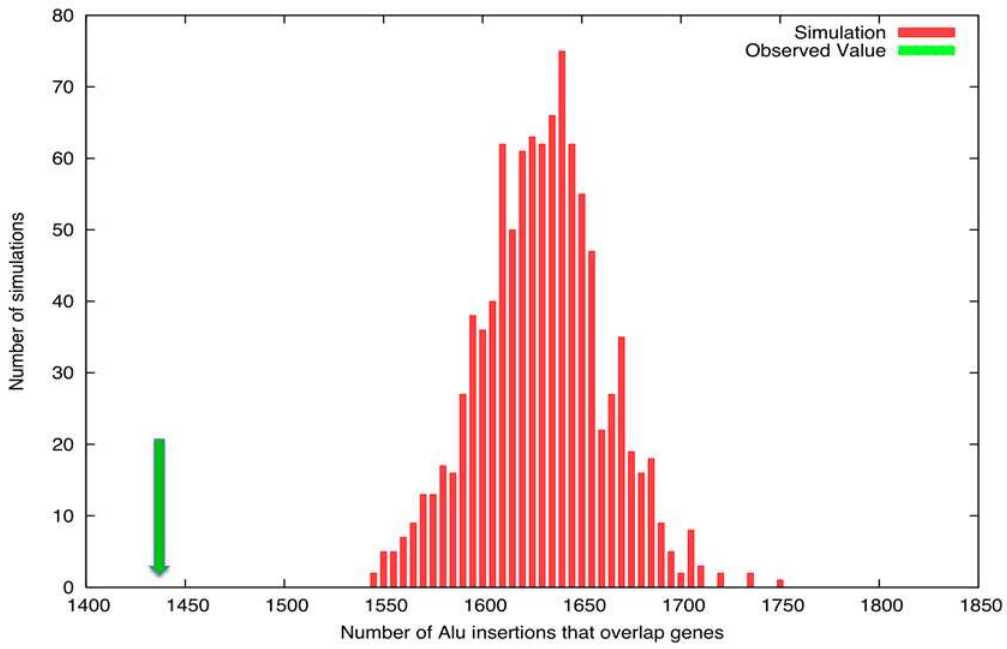


Figure 4.3: Gene overlap analysis. 1437/4342 (33.1%) of predicted Alu insertions map within a human gene as defined by RefSeq (green arrow). The histogram shows the expected distribution of gene overlap based on 1000 permutations.

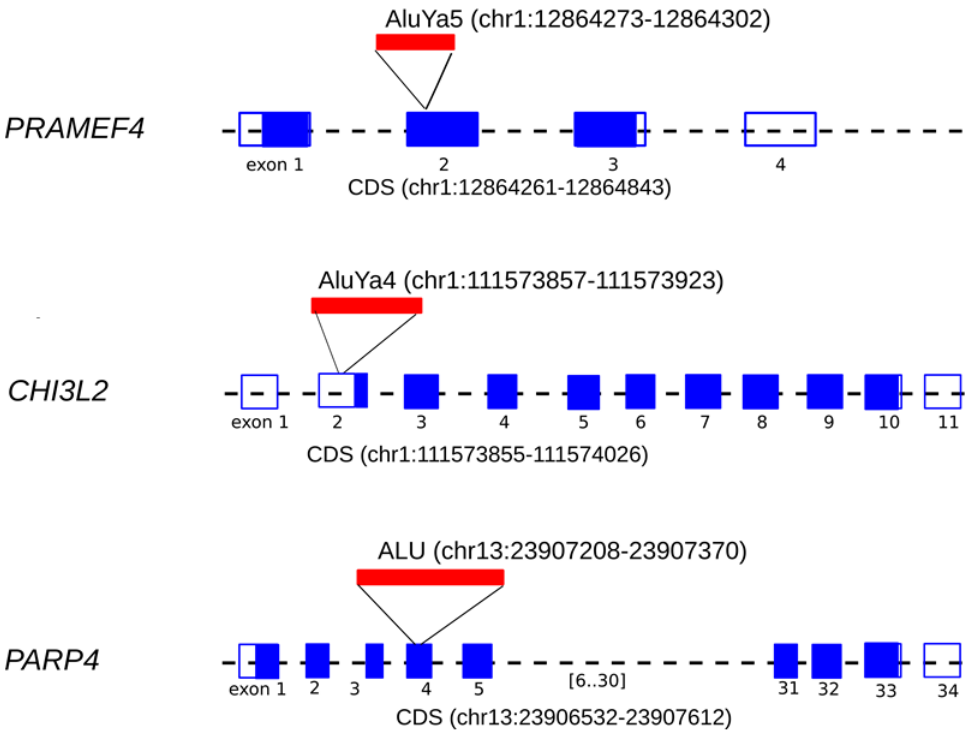


Figure 4.4: Gene disruptions. The locations of three novel insertions within the coding exons of PRAMEF4 (chr1:12,864,273-12,864,302), CHI3L2 (chr1:111,573,857-111,573,923), and PARP4 (chr13:23,907,208-23,807,370) are shown. Unfilled black rectangles represent the exons (and parts of exons) in the untranslated 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 URT.

Table 4.2: The PCR validation results on a some randomly selection of Alu insertion predictions. Note that false negative are calculated as the number of loci that were predicted to be specific to another individual, yet PCR showed Alu insertion in the specified genome.

Individuals	PCR assays	Predicted Alu insertions	False positives	False negative
NA18506	69	31	0 (0%)	1 (2%)
NA18507	69	30	1 (3%)	4 (10%)
NA18508	69	32	1 (3%)	1 (2%)
YH	40	10	0 (0%)	4 (13%)
AK1	40	10	0 (0%)	1 (3%)
KB1	40	10	0(0%)	4 (13%)
NA10851	40	10	1 (10%)	2 (6%)

predicted Alu insertion event in the original target genome. We re-examined the 10 sites that did not initially validate by designing a second PCR assay. For the second assay, we selected oligonucleotides that map further from the predicted integration site and validated 9 out of 10 of these sites (Please see table 4.2). Our combined results suggest excellent sensitivity (63/64 = 98.4%) for VariationHunter, but also suggest caution in interpreting the map location precision based strictly on in silico mapping (detailed results of the PCR experiments are provided in [56]).

Since the novel Alu insertions we detected could, in principle, represent Alu deletions from the reference genome as opposed to newly retrotransposed events, we attempted to assign each Alu insertion to a particular subfamily based on the presence of diagnostic sequence variants. If the events detected were predominantly new retrotransposition, we would predict that the AluY subfamily would predominate-the only known active Alu subfamily [14]. To perform this classification, we compared Alu anchored sequence reads at each site to consensus sequences corresponding to each of the 31 defined Alu subfamilies (RepBase [70]) and used the minimum genetic distance to assign the Alu to a particular subfamily. In some cases where we could not distinguish between two subfamilies (i.e. they were equally divergent with respect to a consensus), both were reported (e.g. an insertion reported as AluYa5/AluYa8 indicates that we were not able to distinguish between these two subfamilies for that particular insertion). In addition, we were not able to confidently assign 106 Alu insertions to subfamilies. Out of the 4,236 of Alu insertions that we could classify, 3,785 (89.3%) of them belonged to the AluY subfamily with the two most active members (Ya5 and Yb8) ranked at the top. We classified 397 Alu insertions as AluS (9%) and 54 as AluJ (1%). These may represent potential deletions in the reference genome or integrations that arose by endonuclease-independent

pathways as opposed to new retrotransposition events driven by target-site primed reverse transcription (see Figure 4.5).

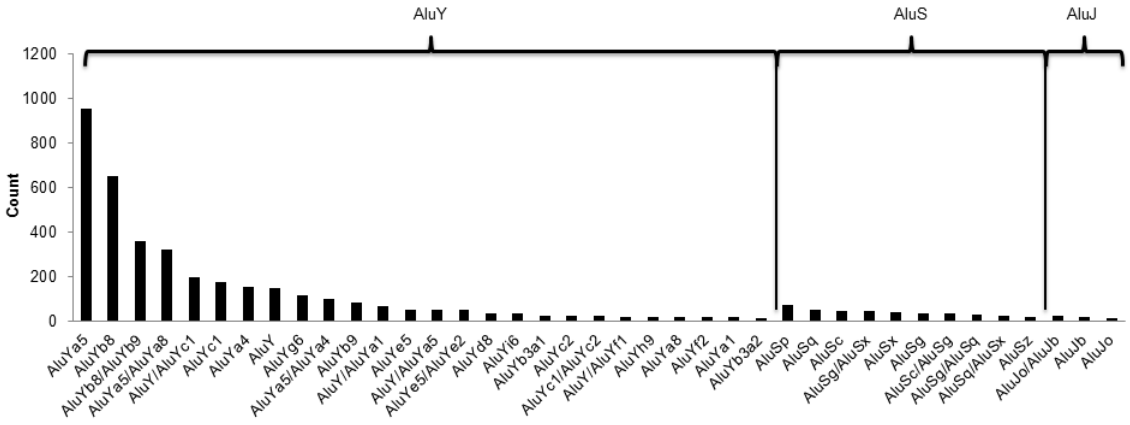


Figure 4.5: Alu frequency distribution among subfamilies. We show the number of predicted Alu integrations in the eight genomes separated by the inferred Alu subfamilies. As expected, AluY class had the highest frequency, where AluJ was the rarest. The Alu classes (AluY, AluS, AluJ) are sorted from youngest to the oldest.

4.2 Familial Transmission

We focused on the parent-child trio (NA18506, NA18507 and NA18058) to assess the transmission characteristics of the novel Alu insertions. We performed two sets of experiments. First, 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 4.6). In the second experiment, we pooled all sequence data from the trio providing approximately 50-fold sequence coverage for each haplotype (Table 4.1). As expected, the analysis led to an increased sensitivity for transmitted Alu insertions, significantly reducing the number of potential de novo candidates. In this analysis, we identified only seven (0.4%) potential de novo insertions out of 1,720 total insertions predicted in the proband (please see Figure 4.6).

We attempted to validate the seven potential de novo insertions but this proved difficult due to the

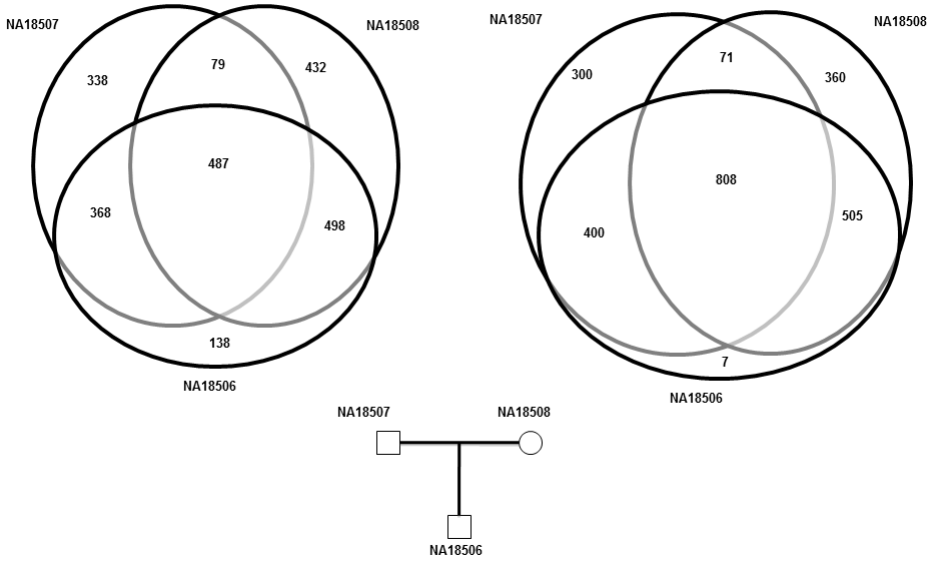


Figure 4.6: Improved specificity in detecting de novo insertions. A three-way comparison of novel Alu insertion polymorphisms in the YRI trio: when they are predicted separately (left Venn diagram), and when the reads from three individuals are pooled together (right Venn diagram). Pooled coverage reduces the number of false-positive de novo events for further consideration.

repetitive nature of sequence flanking the insertion. Despite multiple attempts, we could not design a successful assay for two of the seven predicted events. Of the seven initial de novo predictions, four were confirmed as Alu insertions in the child but found to be transmitted from one of the parents, and the remaining three could not be tested or interpreted.

4.2.1 Genotyping

We have also studied the parent-child trio for homozygous versus heterozygous insertions. Based on our analysis of the parent-child trio, we categorized all NA18506 Alu insertions as homozygous or heterozygous. To extract this information we developed a simple classifier for genotyping Alu insertions that considers two features: 1) the number of concordant paired-end mappings that span the loci of the predicted insertion (y-axis) and 2) the number of discordant paired-end reads that support an Alu insertion (x-axis). Our analysis shows that heterozygous and homozygous genotypes are accurately classified using this simple classifier.

We use a simple linear separator classifier based on two features to genotype the Alu insertions predicted in the donor genome. The first feature is the total number of discordant paired-end reads that support the Alu insertion, and the second feature is the total number of concordant paired-end read mappings that span the insertion locus (if the concordant paired-end read has multiple mappings, a mapping location with the least edit distance is considered). Note, if we assume no paired-end read is mapped incorrectly, we expect that for a heterozygous Alu insertion the total number of paired-end reads spanning the insertion locus will be almost half of the total paired-end reads that support the Alu insertion. On the other hand, if we assume the insertion is homozygous, we then expect that the total number of concordant paired-end mappings that span the locus to be almost zero. We use a two-dimensional space to represent these insertions, where the x-axis is the number of discordant paired-end reads supporting the Alu insertion and the y-axis is the number of concordant paired-end reads that span the locus. In this two-dimensional space, the heterozygous insertions should fall close to the line $y = 1/2x$ and the homozygous insertions should be close to $y = 0$. Thus, we can easily classify insertions based on their distance to these two lines. This is equivalent to using the line $y = 1/4x$ as the separator between these two classes. We experimentally tested the genotyping results of 29 previously validated Alu insertions in the YRI trio using PCR, where 28 of the 29 insertion polymorphisms were correctly genotyped (Figure 4.2.1). The only locus incorrectly genotyped in NA18506 (chr13: 78,169,5920-78,169,605) was also the only locus incorrectly genotyped in the NA18508 and NA18507 genomes (see Figure C.1(a), C.1(b)

in Appendix C). One possible explanation may be that the region is enriched for long terminal repeat (LTR) elements and long interspersed nuclear elements (LINEs) in the flanking region confounding detection and validation.

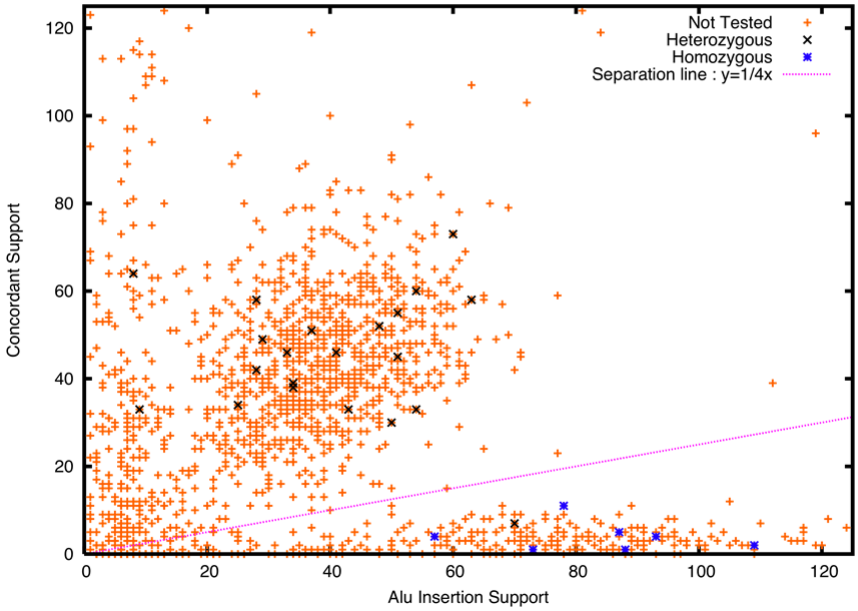


Figure 4.7: The classifier compares the number of reads consistent with the reference genome (concordant) when compared to the number of reads supporting an Alu insertion polymorphism. Experimentally confirmed heterozygous insertion (black dots) are compared to homozygous insertions (blue dots). Simple classifier $y = 1/4x$ can correctly genotype 27/28 loci for NA18506 insertions as homozygous or heterozygous.

4.3 Genome comparisons

We compared the extent of shared Alu insertion polymorphisms among the analyzed genomes in this study (See Figure 4.9). Based on our limited sample size of eight genomes, we found that 50% of these novel Alu insertions were observed in two or more individuals, suggesting an allele frequency $> 10\%$ (Figure 4.8). Due to the non-uniformity in sequence coverage, this is likely an underestimate as a result of false negatives. Therefore, we repeated this analysis, limiting it to four unrelated genomes, each representative of a different human population, namely YH (Han Chinese),

NA18506 (YRI), AK1 (Korean), and NA10851 (CEU). Of the Alu insertions, 4% (137/3446) were shared among all four genomes but were not present in the reference genome (NCBI Build 36). Considering the diversity of the sampled genomes, we conclude that these 137 loci are common to most humans, and the reference genome likely represents a rare polymorphism (Figure 4.9).

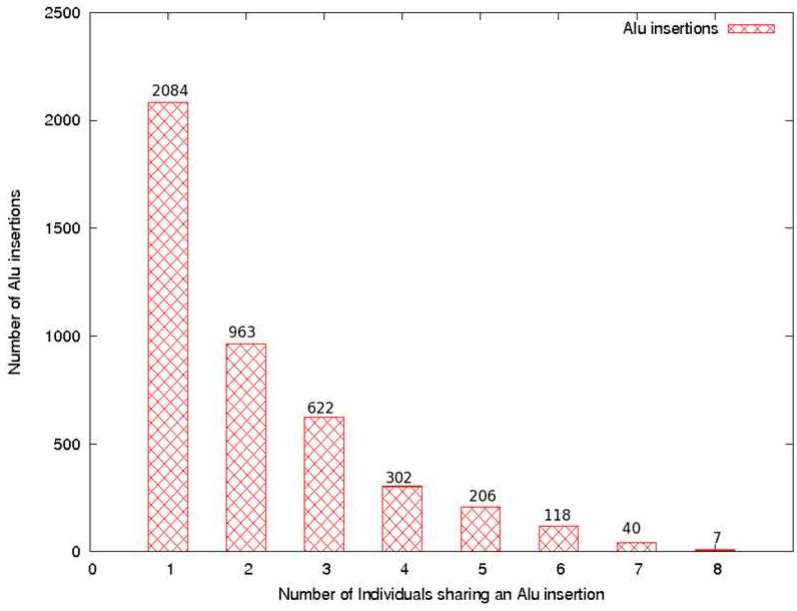


Figure 4.8: The histogram shows the number of Alu insertions that are unique and shared among two or more genomes.

Note that although we find less-common Alu insertions between AK1 and YH than we find between YH and NA10851, we believe this is only an artefact of lower sequence coverage in the AK1 genome compared with the other two genomes. As expected, the YRI genome shows greater genetic diversity. Approximately 59% (1016/1720) of the Alu insertions predicted in NA18506 are unique when compared with the other genomes, where the proportions of unique Alu integration loci in other genomes range between 37% and 45%. We identify 10%-15% more Alu integrations in the YRI samples, even after controlling for differences in sequence coverage. In addition, fewer YRI insertions were previously reported in dbRIP (16% [388/2451] of YRI insertions vs. 20% [488/2430] of non-African insertions). When we also compare with other recently published sets of novel Alu insertions [68, 153] in addition to dbRIP, we find 24% (589/2451) of YRI insertions and

31% (744/2430) of non-African insertions were previously reported.

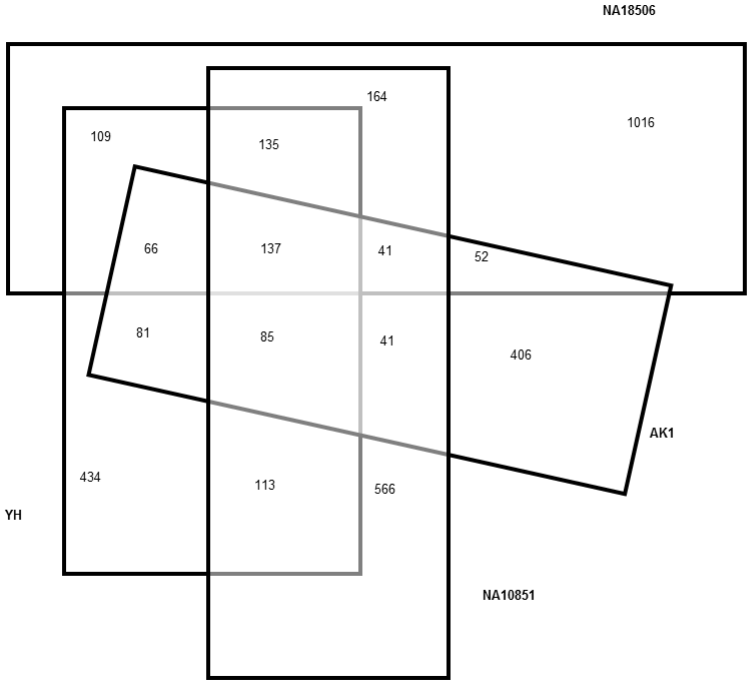


Figure 4.9: A Venn diagram of shared novel Alu insertions among four genomes. The African (NA18507) genome shows the greatest number of individual-specific Alu integration polymorphisms when compared with the Asian (AK1 and YH) and European (NA10851) genomes. This effect holds even after correcting for differences in genome coverage

Chapter 5

Simultaneous Structural Variation Discovery

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 this chapter we propose to move away from this two-step approach to a novel one in which all genomes are compared with the reference genome simultaneously for obtaining much higher accuracy in structural variation detection. For this purpose, we introduce the maximum parsimony-based simultaneous structural variation discovery problem for a set of high throughput sequenced genomes and provide efficient algorithms to solve it. We compared the proposed framework with the conventional framework, on the genomes of two trio (the Yoruban mother-father-child trio and the CEU mother-father-child trio) We observed that the conventional framework predicts an unexpectedly high number of de novo variations in the child in comparison to the parents and misses some of the known variations. Our proposed framework, on the other hand, not only significantly reduces the number of incorrectly predicted de novo variations but also predicts more of the known (true) variations.

5.1 Introduction

High throughput - next generation sequencing (NGS) technologies are reducing the cost and increasing the world-wide capacity for sequence production at an unprecedented rate. Large scale projects based on NGS aim to sequence 2000 [1] or many cancer genomes [67] and analyze genomic variation at a population scale. Genomic variation, especially structural variation (involving insertion, deletion, duplication, translocation and transposition events) detection through NGS promises to be one of the key diagnostic tools for cancer and other diseases with genomic origin [41], [138]. One recent study [83] for example, demonstrates that patient-specific structural variants identified in blood samples could be used as personalized biomarkers for monitoring tumor progression and response 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. [28] or [148]). Although it is possible that such genomic signals simply do not exist in the cancer types studied, another likely explanation is that the computational tools used in these studies were not sufficiently accurate to correctly identify and/or prioritize recurrent structural variants. The emerging area of personalized genomics will surely benefit from computational tools that can correctly identify recurrent structural variants among a collection of genomes and transcriptomes.

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 chapter 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 [1, 107]), which include WGSS data from diverse populations (and members of the same population or even family).

Existing Methods for Structural Variant (SV) Discovery and Their Limitations. 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. PEMer [78] for example, maps each paired- end read to a unique location through the mapping software MAQ [84]. A number of followup studies that employ a similar ”hard clustering” approach [101] include Pindel [156] and BreakDancer [27], all focus only on the ”best mapping” of each read, provided by the mapping software in use. A survey by Medvedev et al.[101] summarizes the basis of decision making for each of these methods and report briefly on their performance. 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 recent paper [46], it was demonstrated that by ignoring possible mapping locations of a read may lead to significant loss of accuracy in structural variation detection. 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, 4, 85] have been introduced in the past two years. Here, paired-end reads are mapped to all potential locations - through the use of the mapping algorithms such as mr and mrs *FAST* [4, 46]. 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 [101]. VariationHunter [55] (please see chapter 3) 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, deletion and inversion polymorphisms. MoDiL [85], as well as its followup MoGUL [86] evaluate the clusters of reads that seem to indicate a structural variant using a probabilistic framework, while Hydra [118] uses heuristics (based on the algorithmic strategies of VariationHunter) to detect structural variant breakpoints in the mouse genome. MoGUL [86] focuses on finding common insertion and deletion events in a pool of multiple low coverage sequenced genomes.

In this chapter we demonstrate, for example, that on the well known NGS genomes of the Yoruban family (involving a child, the mother and the father, NA18506, NA18507, NA18508), the independent application of VariationHunter (one of the few 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 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 by using the reference 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 (e.g. genetic relationship) of the individual donors sequenced, 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 chapter 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 [1, 107]. We compare the Alu insertion predictions with Alu polymorphism loci reported in dbRIP [149]. In both cases we observe that CommonLAW provides a 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 [1]. We demonstrate the predictive power of CommonLAW by comparing its calls with the validated deletions reported in [1, 107].

5.2 Methods

5.2.1 Simultaneous Structural Variation Discovery among Multiple Genomes

Given a reference genome (e.g. human genome build36) 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

[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.

event in G_k [144, 147]. 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 chapter 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 in Hormozdiari et al. [55] (part of Chapter 3) with the introduction of novel approximation algorithms for the "Maximum Parsimony Structural Variation" (MPSV) discovery problem. In chapter 3, an 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 [12, 137, 55, 58, 47]. 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, 58, 47].

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.

5.2.2 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 Chapter 3.

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. 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 - as discussed in the Introduction section.

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 at least one discordant read is 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. 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.

5.2.3 Complexity of SSV-MG problem

It is easy to see that 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 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.)

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 [5] 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 4. *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. [5]. 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 5. *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

[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.

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 [5] for a small constant c.

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

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

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. [5], for a sufficiently small constant c.

□

5.2.4 An 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 [145] 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 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.

5.2.5 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 multicolored 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 [72])

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.

Our additional post-processing 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 multicolor subsets. Figure 1 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.

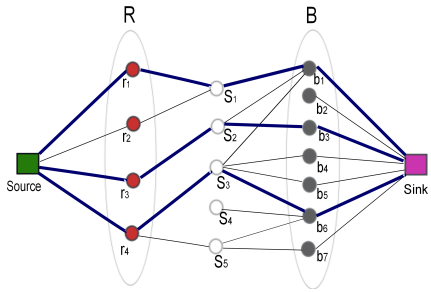


Figure 5.1: The set $R = \{r_1, r_2, r_3, r_4\}$ represents the red elements and $B = \{b_1, b_2, \cdots, b_7\}$ represents the black elements. The selected subsets are $S_1 = \{r_1, r_2, b_1\}, S_2 = \{r_3, b_1, b_3\}, S_3 = \{r_4, b_1, b_4, b_5, b_6\}, S_4 = \{b_6\}, S_5 = \{r_4, b_6, b_7\}$. All the 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.

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

It is possible to further improve the algorithms presented in section 5.2.4 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]

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

[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.

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 [145] gives a $2 \cdot \frac{\omega_{\max}}{\omega_{\min}}$ approximation.[5] The following algorithm achieves a better 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 $1 + \frac{\omega_{\max}}{\omega_{\min}}$.

Theorem 6. *Red-Black-Assignment-F2 problem can be approximated within a factor of* $1 + \frac{\omega_{\max}}{\omega_{\min}}$.

Proof. We remind the reader that the instance of Red-Black-Assignment-F2 problem is denoted by H and a maximal matching in H is denoted by M. $R = \{r_1, r_2, \cdots, r_p\}$ is the set of red edges and $B = \{b_1, b_2, \cdots, b_q\}$ is the set of black edges in M.

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

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

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

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.

□

5.2.7 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 3.3 and [58]) 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 [145] 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 $O(\frac{\omega_{\max}}{\omega_{\min}} \log n)$.

We have also applied a very similar algorithm to SSC-W, in context of gene fusion discovery. The algorithm uses data from two sources of RNA-Seq and WGSS data simultaneously to predict gene fusions with high accuracy [100] (the algorithm is called Comrad).

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. In section 3.3 (Hormozdiari et al. [58]) we introduced a set of mathematical rules to prevent selecting SV events that can not 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 3.3 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 section 3.3 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 cost-effectiveness of SV cluster s in iteration i is $\frac{w_{s'_i}}{|s'_i|}$, where s'_i is the subset

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

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.

5.3 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 [18]). The second trio is a family from Utah with European ancestry (CEU: NA12878, NA12891, NA12892) sequenced with high coverage by the 1000 Genomes Project [1]. We aligned the downloaded paired-end reads to the human reference genome (NCBI Build 36) using *mrFAST* [4]. Statistics for each dataset are provided in table 5.1 (after removing low-quality paired-end reads).

Table 5.1: 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	40.1x	255x
NA18507	YRI	2.261×10^9	36-41bp	208bp	27.1x	157x
NA18508	YRI	3.175×10^9	35bp	203bp	37x	214x
NA12878	CEU	1.049×10^9	36-76bp	201bp	32.3x	70x
NA12892	CEU	0.510×10^9	35-51bp	153bp	12.6x	26x
NA12891	CEU	0.551×10^9	35-51bp	148bp	13.8x	27x

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 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 (section 3.1.2) for deletions and VariationHunter for Alu insertions. [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 mobile element insertions.

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

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

5.3.1 Mobile element insertions

We previously described a method for using VariationHunter to discover mobile element insertions, such as Alu elements, given a reference genome and WGSS data from a single donor genome[58, 56]. We sought to compare the ISV&M approach, with an SSV-MG approach using SSC and SSC-W approximated algorithms, in order to establish the benefits of SSV-MG approach for mobile element insertion discovery.

We applied the ISV&M and the two SSV-MG approaches (SSC and SSC-W) 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[149] - which would provide an estimate of the trade-off 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, we believe that finding a match in dbRIP

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

is a good indicator that the prediction is a true positive. (This is depicted in Figure 5.2(a), which provides an indirect comparative measure for the true positive/false negatives rate as a function of the calls made.) We consider a predicted Alu insertion to match an Alu insertion loci reported in dbRIP [149] if the reported loci in dbRIP are 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 on paired-end read support. The total number of dbRIP match Alu insertions is consistently higher for SSC and SSC-W methods in comparison to ISV&M across the range of thresholds Figure 5.2(a).

We expect to see many common Alu insertions in all three genomes (since they are members of a family from the same population); ISV&M only predicted 507 common Alu insertion loci in all three individuals, while SSC predicted 1044 and SSC-W predicted 1257 (See figures 5.2(b), 5.2(c), and 5.2(d)).

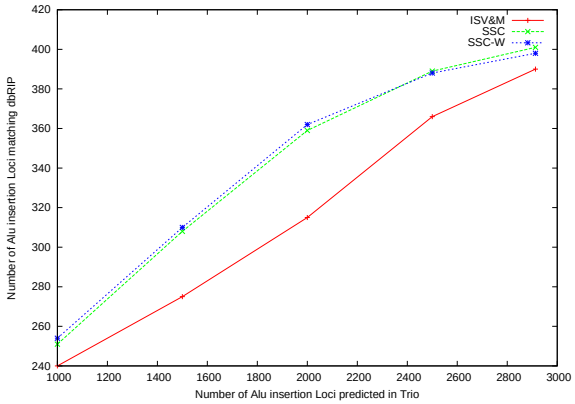
The rate of de novo Alu insertions is estimated at one new Alu insertion per 20 births [30]. Thus, it is quite unlikely that the genome of the child (NA18506) in the YRI trio contains a significant number of Alu insertions not present in the parent genomes. However, an ISV&M analysis using VariationHunter, 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 majority of these 410 events are likely misclassified as de novo event by ISV&M. Interestingly, using the SSC algorithm, this number was reduced to only 20 de novo events among the top 3000 predictions (figure 5.2(c)). Note that using the SSC-W algorithm [9] number of de novo Alu insertions was reduced to zero in the top 3000 Alu insertion predictions (see figure 5.2(d)).

5.3.2 Deletions

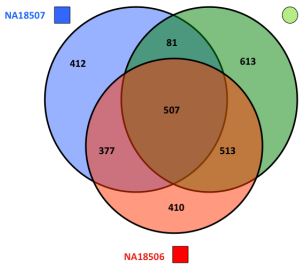
In this section we have compared the deletion calls made by algorithms proposed for the SSV-MG approach (SSC-W and SSC-W-CR) with the ISV&M approach (VariationHunter [55] was used for this framework). We predicted medium to large size deletions ($> 100bp$ and $< 1M$bp) in both YRI and CEU trios using the ISV&M and SSV-MG approaches. The validated SV events reported in the recent study of the 1000 Genomes Project Consortium [107] were used to test the quality of the predictions in the CEU and YRI trio. The results of this comparison are shown in table 5.2. To make the comparison as through as possible, we have considered different subsets of calls for each

[8] The 3000 loci with highest number of paired-end reads support were considered for each method

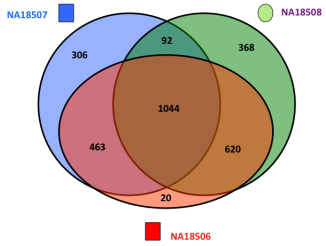
[9] the weights used for SSC-W were derived from the fraction of Alu insertion common between the individuals reported by SSC results



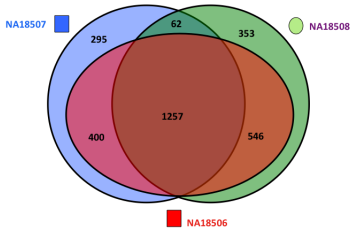
(a) Alu insertion loci prediction and comparison with dbRIP



(b) ISV&M



(c) SSC



(d) SSC-W

Figure 5.2: Alu insertion analysis of the YRI trio genomes. Figure (a) compares 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 total number of Alu insertions, while y-axis represents subset of them which number of match dbRIP. Each data point in this plot was obtained by sorting each set of predictions based on its support level and selecting the top subset of predictions. Figures (b), (c) and (d) detail the number of common and de novo events in each genome as predicted by the ISV&M, SSC and SSC-W algorithms respectively (the top 3000 predictions were considered).

method (using different thresholds on number of predictions considered for each method) [10]. The SSV-MG algorithms, consistently produced more known positive predictions, in the YRI and CEU trio, compared to the ISV&M approach (see table 5.2).

We have also compared the deletions predicted for CEU trio by ISV&M, SSC-W and SSC-W-CR algorithm with the validated calls reported in recent study by Mills et al. [107] for each individual (see table 5.3).

Table 5.2: 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 [107]. The loci of a reported deletion should be in the range of 300bp from a loci reported in [107] 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 [107] (from 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 [107].

We also studied the number of de novo deletions reported in the child genome of CEU trio (NA12878) as predicted by each method. Similar to Alu insertions, we do not expect a significant number of de novo deletions in the child genome. Among the top 5000 deletion loci predicted by the ISV&M approach for CEU trio, 84 were predicted to be de novo events (with high probability majority of them are misclassified as de novo). However, among the top 5000 deletion loci reported by the SSC-W algorithms 34 were predicted to be de novo. This is a reduction of more than 50% on number of misclassified deletions as de novo event.

[10] For each threshold c on total number of predictions considered by each method, we sorted deletions by their support (number of paired-end reads supporting the deletion) and selected the top c deletions

Table 5.3: 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 top 5000 deletion predictions for the CEU trio by the three approaches were considered in this comparison. These set of calls were compared to the validated deletion for each of the three individuals reported in [107].

Chapter 6

Conclusion and Discussion

In this dissertation we have considered problems for genome resequencing and mainly structural variation discovery using high-throughput sequencing technologies. As mentioned in the first chapter of this thesis, the advent of high-throughput sequencing has completely changed the landscape of genomics. Thus, development of efficient algorithms and tools for genome analysis using HTS data has become crucial.

In this thesis we mainly consider algorithms for structural variation discovery using high-throughput sequencing technologies. We adopted the paired-end read mapping strategy for the structural variation discovery problem, with a consideration toward new challenges arising as a result of new sequencing technologies (especially the fact that shorter reads are produced by the new sequencers). The shorter reads result in ambiguous mapping of some paired-end reads, thus prediction of structural variation becomes more challenging from a computational standpoint.

In chapter 2 of this dissertation we have considered the problem of designing a high-throughput sequencing experiment where we want to sequence many genome fragments (BACs, fosmids, or captured exons of interest). The goal is to pool these fragments for sequencing such that the complications arising as a result of repetitions are reduced. We have shown that our method of pooling these fragments reduces the conflicts (repetition) between fragments in the same pool by 50% in comparison to random pooling.

In chapter 3 of this thesis we introduced our model for structural variation discovery using HTS data. We model the SV discovery problem using paired-end data as a combinatorial optimization problem, where the goal is to minimize the total number of structural variations predicted (maximum parsimony approach). We provided an approximation algorithm for this problem and developed a tool known as VariationHunter, which is able to discover structural variation ranging from small

insertion, deletion, inversion, and transposition. We also introduced a weighted version of VariationHunter, where not only structural variations have weights but also paired-end read mappings picked contribute to the final objective function. The VariationHunter algorithm is shown to be efficient and reliable in practice.

We also provide a description of our computational pipeline, *NovelSeq*, which is able to efficiently assemble the novel sequence insertions and build maps of insertions by anchoring the sequences back into the reference genome assembly. This pipeline also uses the same objective function as VariationHunter and is also explained in chapter 3 of this dissertation.

Finally, in chapter 3 we extend our maximum parsimony framework to consider the "conflicts" between different possible variations in the donor genome. Our extended VariationHunter, which is denoted as VariationHunter-CR, is now capable of resolving incompatible SV calls through a conflict resolution mechanism that no longer requires post-processing heuristics. 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 human individuals, as well as non-human species.

In chapter 4 of this thesis, we used VariationHunter for predicting *Alu* insertions in eight different individuals. We predicted more than 4000 loci of Alu insertion on the human genome in comparison to the reference genome. It should be noted that we selected a subset of these Alu insertions for validations using PCR experiments and more than 95% of these predicted Alu insertions were validated.

Finally in chapter 5 of this thesis we demonstrated that analyzing a collection of high-throughput sequenced genomes jointly and simultaneously improves structural variation discovery, especially among highly related genomes. We focus on discovering deletions and *Alu* insertions among high-throughput paired-end sequenced genomes of family members and show that the algorithms we have developed for simultaneous genome analysis provide lower false positive rates in comparison to existing algorithms that analyze each genome independently. Our algorithms, which are collectively called CommonLAW (Common Loci Structural Alteration Widgets), aim to solve the maximum parsimony structural variation discovery for multiple genomes optimally through a generalization of the maximum parsimony formulation and the associated algorithms introduced in chapter 3 for a single donor genome. We believe that the CommonLAW framework will help faciliate studies of 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.

6.1 Future Works

Major progress has been made on the methods for structural variation discovery in the past few years; however, the algorithms still need to be improved significantly. Here we describe several potential directions in which our algorithms can be extended.

First of all as it can be seen in the results section of chapter 3, VariationHunter and VariationHunter-CR are not able to find all the validated deletions reported in Kidd et al. [73]. More precisely there are still around 30% of validated deletions reported in Kidd et al. [73] (for the same individual) which we were not able to discover using VariationHunter with Illumina data. To understand the reason behind this, we did a simple experiment. We looked at the validated deletions reported in [73] which VariationHunter did not find (using Illumina data) and tried to match them with the available discordant paired-end read mappings.

Interestingly, almost none of those deletions had any discordant paired-end read mapping support. This observation indicates that the general framework that discards paired-end reads which have a concordant mapping is probably resulting in a reduction of true positive rate. A new method which utilizes the concordant paired-end reads to find those very hard to find structural variations is indeed desired.

Second of all, based on the 1000 Genomes Project results [107], we know that around 20% of the deletion calls made by VariationHunter are not correct. One way to try to improve the false discovery rate is by using other signatures available (such as read depth or split read) to prune SVs, which are less likely. For instance, we can modify the VariationHunter-CR algorithm to have nodes in the conflict hypergraph for regions with a high depth of coverage indicating no deletion in those regions. We have experimented with this idea (using simple extensions of VariationHunter-CR), and it seems to improve the false discovery rate significantly. We believe a theoretical extension to VariationHunter-CR which considers depth of coverage as an additional signature can significantly improve the results.

Third of all the algorithms presented for structural variation discovery in this thesis are based on the maximum parsimony objective function. However, we can also study this problem under a maximum likelihood or maximum a posteriori objective function and try to optimize them using expected maximization (EM) or the Monte Carlo method. It should be noted that we have already developed a probabilistic method to calculate the probability of each structural variation (the method

was presented in [55]). In the probabilistic method proposed in [55], a set of non-linear equations were provided and a heuristic method for calculating the probability of each structural variation solely based on read mappings was used.

Fourth of all the performance of simultaneous structural variation discovery algorithms (SSV-MG) provided in chapter 5 is dependent on the homogeneity of sequencing depth of coverage across all genomes. An algorithm which is not oblivious to the differences between depth of coverage of genomes sequenced is needed.

Finally, third generation sequencing technologies are being developed. These technologies (such as Pacific Biosciences, Oxford Nanopore or Ion Torrent) promise a much longer read length than the second generation of sequencing technologies; however, their projected throughput is less than the second generation (at least for the coming few years). We suggest developing algorithms which can use both the low coverage, long reads data produced by third generation machines in combination with high coverage short reads of second generation machines to discover SVs. We need new problem formulation and algorithm developments to be able to use such a hybrid set of data.

Appendix A

Conflicting Valid Clusters for Haploid Genome

For each SV cluster, we define a segment in the reference genome as the *conflict zone* of the SV cluster and assume that two different SV clusters should not overlap in their conflict zones in a *haploid* genome sequence (or the two clusters will be in conflict).

In order to define the conflict zone of a SV cluster, we will first give some definitions. For each SV cluster $VClu$, we define $minL(VClu)$, $maxL(VClu)$, $minR(VClu)$, $maxR(VClu)$ as following:

$$\begin{aligned} minL(VClu) &= min\{L_l(a_jpe_i)|a_jpe_i \in VClu\} \\ maxL(VClu) &= max\{L_r(a_jpe_i)|a_jpe_i \in VClu\} \\ minR(VClu) &= min\{R_l(a_jpe_i)|a_jpe_i \in VClu\} \\ maxR(VClu) &= max\{R_r(a_jpe_i)|a_jpe_i \in VClu\} \end{aligned}$$

Next, for each valid cluster $VClu$ supporting different types of variations, we define the conflict zone $CZ(VClu)$. Not that event *Copy from left* indicates a transposition event where the copied segment is pasted to its right (i.e. $Pos_{Br} > Pos_R$), and event *Copy from right* indicates a transposition event where the copied segment is pasted to its left (i.e. $Pos_{Br} < Pos_L$). The set of rules for calculating $CZ(VClu)$ is as follow:

$$\text{CZ(Vclu)} = \begin{cases} [minL(VClu), maxL(VClu)] & \text{Deletion} \\ [minL(VClu), maxL(VClu)] & \text{Insertion} \\ [minL(VClu), maxL(VClu)] \cup [minR(VClu), maxR(VClu)] & \text{Inversion} \\ [minR(VClu), maxR(VClu)] & \text{Copy from left} \\ [minL(VClu), maxL(VClu)] & \text{Copy from right} \end{cases}$$

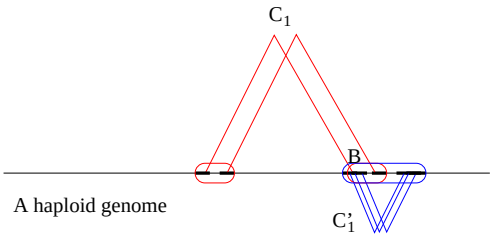
Note that the conflict zone of an SV cluster which supports an inversion is split to two non-overlapping conflicting zones. The way we define two clusters to be in conflict in haploid genome is based on their conflict zones. Cluster $VClu_i$ and $VClu_j$ are in conflict if the $ZC(VClu_i)$ and $ZC(VClu_j)$ intersect. Note that, in the case that both clusters $VClu_j$ and $VClu_i$ support an inversion and their conflict zones have no intersection still they can be in conflict if they have a "zig-zag" pattern. Two inversion clusters $VClu_i$ and $VClu_j$ have a "zig-zag" pattern if these two conditions hold:

1) segment $[minL(VClu_i), maxL(VClu_i)]$ false inside of segment $[minL(VClu_j), maxR[(VClu_j)]$ and

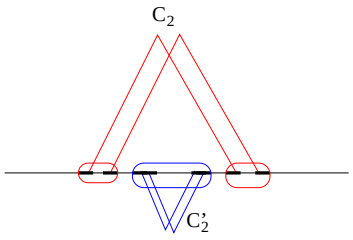
2) segment $[minR(VClu_i), maxR(VClu_i)]$ false outside of segment $[minL(VClu_j), maxR(VClu_j)]$.

In Figure A.1(a), two different SV clusters [1], C_1 and C_1' are seen where C_1 represents an *inversion* occurring in a block of a haploid genome sequence and C_1' a *insertion*. Let B (see Figure A.1(a)) be a block where the *ends* of the mate-paired reads suggesting the inversion event location. As it can be seen, in Figure A.1(a), C_1 and C_1' are conflicting SV clusters since both of them cannot occur at the same time. Figure A.1(b) presents two SV clusters, C_2 and C_2', one representing an insertion and the other one representing an inversion. In this case, C_2 and C_2' can indeed be valid SV clusters at the same time and are not conflicting with each other. Another example is shown in Figure A.1(c) where SV clusters C_3 and C_3' represent an insertion and a deletion event and conflict with each other.

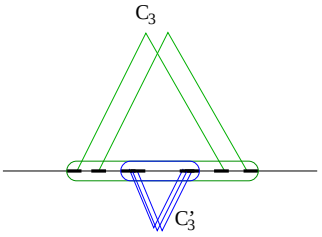
[1] We remind the reader that each SV cluster represents a set of discordant mate-paired reads supporting exactly one particular SV.



(a) Conflicting: C_1 is an inversion cluster, shown in red together with its conflict zone, and C_1' is an insertion cluster, shown in blue.



(b) Not Conflicting: C_2 is an inversion cluster and C_2' is an insertion cluster.



(c) Conflicting: C_3 is a deletion cluster and C_3' is an insertion cluster.

Figure A.1: (a) shows two SV clusters together with their conflict zones. The conflict zones share a common block B in a haploid genome sequence; thus we consider them as conflicting SV clusters. (b) C_2 and C_2' support an inversion event and an insertion event. Their conflict zones are not intersecting; thus C_2 and C_2' are not considered conflicting. (c) shows two SV clusters that are conflicting in a haploid genome sequence, since $CZ(C_3) \subset CZ(C_3')$.

Appendix B

Alu insertions in genes and exons

Table B.1: Exon overlap predictions

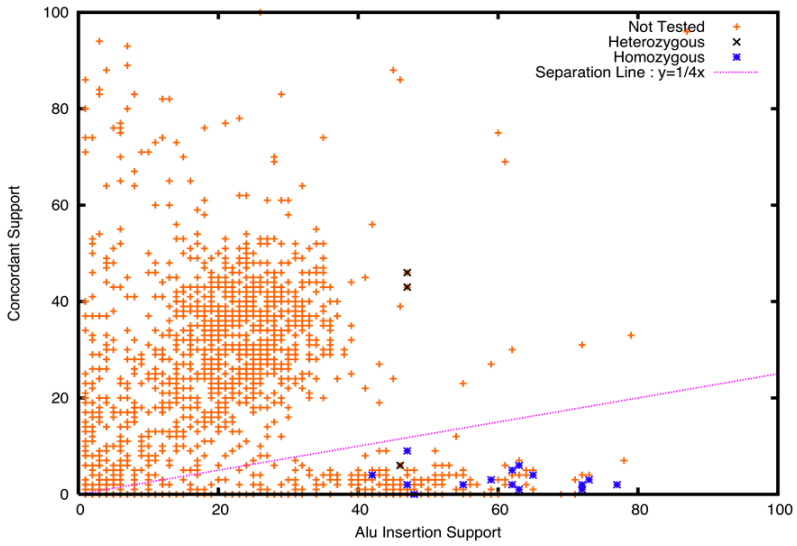
Chro	Start	End	Strand	Subfamily	NA18506	NA18507	NA18508	YH	KB1	AK1	NA10851	HGDP01029	Gene	Exon(start)	Exon(End)	Completely inside
chr1	12,864,273	12,864,302	+	AluYa5	no	yes	no	yes	no	no	no	no	PRAMEF4	12864261	12864843	TRUE
chr1	36,247,208	36,247,329	-	AluYa5	no	no	no	yes	no	yes	no	no	EIF2C3	36247084	36247232	FALSE
chr1	44,076,396	44,076,491	+	AluYa8	yes	no	yes	no	no	no	no	no	ST3GAL3	44076477	44076570	FALSE
chr1	111,573,857	111,573,923	+	AluYa4	yes	yes	no	no	no	no	no	no	CHI3L2	111573855	111574026	TRUE
chr1	179,124,119	179,124,216	+	AluYa5	no	yes	no	no	no	yes	no	no	XPR1	179119764	179126036	TRUE
chr1	231,585,747	231,585,827	+	AluYa5	yes	yes	yes	no	no	yes	no	no	KIAA1804	231584673	231587517	TRUE
chr11	7,049,317	7,049,405	+	AluYb8	yes	yes	yes	yes	no	yes	yes	no	NLRP14	7048979	7049333	FALSE
chr11	43,833,997	43,834,050	-	AluYa5	yes	yes	yes	no	no	yes	no	no	HSD17B12	43833269	43834745	TRUE
chr11	87,879,791	87,879,792	+	AluYa5	yes	yes	yes	no	no	no	no	no	GRM5	87877392	87882320	TRUE
chr12	95,871,446	95,871,487	-	AluYa5	no	yes	no	no	no	no	no	no	NEDD1	95869857	95871592	TRUE
chr13	23,907,208	23,907,370	+	ALU	no	no	no	yes	no	no	no	no	PARP4	23906532	23907612	TRUE
chr15	89,638,841	89,638,922	-	AluYb8/9	no	no	no	no	no	no	no	yes	SV2B	89636602	89639654	TRUE
chr16	73,500,969	73,501,211	+	AluSx	yes	yes	yes	no	no	no	no	no	WDR59	73500899	73501056	FALSE
chr17	3,748,988	3,749,251	-	AluJb	no	no	no	yes	no	no	no	no	P2RX1	3748976	3749067	FALSE
chr17	35,550,133	35,550,142	-	AluSg	no	no	no	no	no	yes	no	no	CASC3	35550032	35550558	TRUE
chr17	44,727,722	44,727,815	-	AluYb8	no	no	no	no	no	no	yes	no	ZNF652	44721567	44731285	TRUE
chr17	68,846,663	68,847,077	-	AluJo/Jb	no	no	no	no	no	no	yes	no	SDK2	68842118	68846674	FALSE
chr18	15,315,613	15,316,164	-	AluY/AluYc1	no	no	no	no	no	no	yes	no	LOC644669	15315659	15315729	FALSE
chr19	7,442,856	7,442,883	-	AluYa5	no	no	no	yes	no	no	no	no	ARHGEF18	7441422	7443370	TRUE
chr19	42,723,257	42,723,369	-	AluYa4	no	no	no	no	no	no	yes	no	ZNF793	42719638	42726079	TRUE
chr2	208,182,733	208,182,758	-	AluSg/Sx	yes	yes	yes	no	no	no	no	no	FAM119A	208182083	208186412	TRUE
chr20	1,095,953	1,095,981	-	AluYb9	no	yes	no	no	no	no	no	no	PSMF1	1093672	1096426	TRUE
chr20	1,494,054	1,494,264	+	AluYa5	yes	yes	yes	yes	no	no	yes	yes	SIRPB1	1493029	1494143	FALSE
chr3	15,269,325	15,269,328	+	AluYa5	yes	yes	yes	no	no	no	no	no	CAPN7	15267626	15269427	TRUE
chr3	172,801,992	172,802,000	+	AluYa5	yes	no	yes	no	no	no	no	no	PLD1	172801311	172803786	TRUE
chr4	186,598,815	186,598,959	+	AluYa5	yes	yes	yes	yes	no	yes	yes	no	C4orf47	186598687	186598867	FALSE
chr5	42,755,689	42,755,706	-	AluYe5	no	no	no	yes	no	no	no	no	GHR	42754311	42757683	TRUE
chr5	61,892,816	61,892,936	+	AluYb8	yes	yes	no	no	yes	no	yes	no	IPO11	61892719	61892841	FALSE
chr5	95,019,842	95,020,026	-	AluYa4	yes	yes	yes	yes	no	yes	yes	no	SPATA9	95019775	95020373	TRUE
chr6	30,000,810	30,000,856	+	AluYb8	yes	no	yes	yes	no	no	no	no	HCG4P6	30000347	30001407	TRUE
chr6	31,546,934	31,547,115	-	AluYe5/e2	no	no	yes	no	no	no	yes	no	HCG26	31546984	31548163	FALSE

Table B.2: Coding Exon overlap predictions

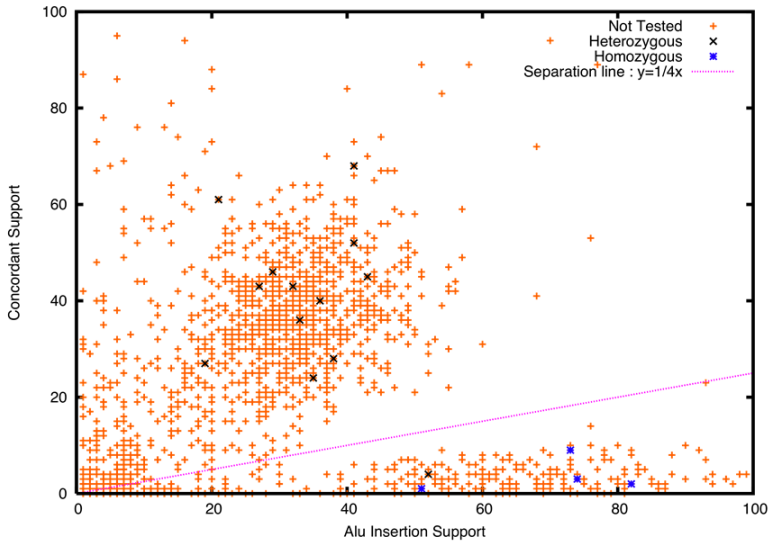
chrom	start	end	Support	Strand	Subfamily	NA18506	NA18507	NA18508	YH	KB1	AK1	NA10851	HGDP01029	Gene	CDS (start)	CDS (end)	Completely inside
chr1	12,864,273	12,864,302	32	+	AluYa5	no	yes	no	yes	no	no	no	no	PRAMEF4	12864261	12864843	TRUE
chr1	36,247,208	36,247,329	10	-	AluYa5	no	no	no	yes	no	yes	no	no	EIF2C3	36247084	36247232	FALSE
chr1	44,076,396	44,076,491	11	+	AluYd8	yes	no	yes	no	no	no	no	no	ST3GAL3	44076477	44076570	FALSE
chr1	111,573,857	111,573,923	15	+	AluYa4	yes	yes	no	no	no	no	no	no	CHI3L2	111573855	111574026	TRUE
chr13	23,907,208	23,907,370	8	+	ALU	no	no	no	yes	no	no	no	no	PARP4	23906532	23907612	TRUE
chr16	73,500,969	73,501,211	14	+	AluSx	yes	yes	yes	no	no	no	no	no	WDR59	73500899	73501056	FALSE
chr17	3,748,988	3,749,251	3	-	AluJb	no	no	no	yes	no	no	no	no	P2RX1	3748976	3749067	FALSE
chr4	186,598,815	186,598,959	183	+	AluYa5	yes	yes	yes	yes	no	yes	yes	no	C4orf47	186598687	186598867	FALSE
chr5	61,892,816	61,892,936	132	+	AluYb8	yes	yes	no	no	yes	no	yes	no	IPO11	61892719	61892841	FALSE

Appendix C

Alu Insertion Genotyping



(a) NA18507



(b) NA18508

Figure C.1: The classifier compares the number of reads consistent with the reference genome (concordant) when compared to the number of reads supporting an Alu insertion polymorphism. Experimentally confirmed heterozygous insertion (black dots) are compared to homozygous insertions (blue dots). Simple classifier $y = 1/4x$ can correctly genotype all loci for NA18507 and NA18508 insertions as homozygous or heterozygous except one loci.

Bibliography

[1] 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–73, Oct 2010.

[2] Alexej Abyzov, Alexander Eckehart Urban, Michael Snyder, and Mark Gerstein. Cnvnator: An approach to discover, genotype and characterize typical and atypical cnvs from family and population genome sequencing. *Genome Res*, Feb 2011.

[3] Can Alkan, Bradley P Coe, and Evan E Eichler. Genome structural variation discovery and genotyping. *Nat Rev Genet*, Mar 2011.

[4] 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.

[5] Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for -restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.

[6] S. F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Molec. Biol.*, 215:403–410, 1990.

[7] Vineet Bafna and Pavel A. Pevzner. Sorting permutations by transpositions. In *SODA*, pages 614–623, 1995.

[8] 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.

[9] J. A. Bailey, Z. Gu, R. A. Clark, K. Reinert, R. V. Samonte, S. Schwartz, M. D. Adams, E. W. Myers, P. W. Li, and E. E. Eichler. Recent segmental duplications in the human genome. *Science*, 297(5583):1003–7, 2002.

[10] S. Balasubramanian and D.R. Bentley. Polynucleotide arrays and their use in sequencing. Patent WO 01/157248, 2001.

[11] 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. *Am J Hum Genet*, 72(3):578–89, Mar 2003.

[12] 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.

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

[14] M. A. Batzer, C. M. Rubin, U. Hellmann-Blumberg, M. Alegria-Hartman, E. P. Leeflang, J. D. Stern, H. A. Bazan, T. H. Shaikh, P. L. Deininger, and C. W. Schmid. Dispersion and insertion polymorphism in two small subfamilies of recently amplified human Alu repeats. *J Mol Biol*, 247(3):418–27, 1995.

[15] MA Batzer, SS Arcot, JW Phinney, M Alegria-Hartman, DH Kass, SM Milligan, C Kimpton, P Gill, M Hochmeister, AI Panayiotis, RJ Herrera, DA Boudreau, WD Scheer, BJB Keats, PL Deininger, and M Stoneking. Genetic variation of recent Alu insertions in the human populations. *J Mol Evol*, 42:22–29, 1996.

[16] S. Batzoglou, D. B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J. P. Mesirov, and E. S. Lander. Arachne: a whole-genome shotgun assembler. *Genome Res*, 12(1):177–89, 2002.

[17] 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):1159–70, Jun 2010.

[18] 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.

[19] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-approximation algorithm for sorting by reversals. In *ESA*, pages 200–210, 2002.

[20] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. *J. ACM*, 41(4):630–647, 1994.

[21] S. Boissinot, P. Chevret, and A. V. Furano. L1 (LINE-1) retrotransposon evolution and amplification in recent human history. *Mol Biol Evol*, 17(6):915–928, Jun 2000.

[22] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, 13(4):721–731, Apr 2003.

[23] Peter J Campbell, Philip J Stephens, Erin D Pleasance, Sarah O'Meara, Heng Li, Thomas Santarius, Lucy A Stebbings, Catherine Leroy, Sarah Edkins, Claire Hardy, Jon W Teague, Andrew Menzies, Ian Goodhead, Daniel J Turner, Christopher M Clee, Michael A Quail, Antony Cox, Clive Brown, Richard Durbin, Matthew E Hurles, Paul A W Edwards, Graham R Bignell, Michael R Stratton, and P. Andrew Futreal. Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nat Genet*, 40(6):722–729, Jun 2008.

[24] M L Carroll, A M Roy-Engel, S V Nguyen, A H Salem, E Vogel, B Vincent, J Myers, Z Ahmad, L Nguyen, M Sammarco, W S Watkins, J Henke, W Makalowski, L B Jorde, P L Deininger, and M A Batzer. Large-scale analysis of the alu ya5 and yb8 subfamilies and their contribution to human genomic diversity. *J Mol Biol*, 311(1):17–40, Aug 2001.

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

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

[27] 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.

[28] 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, Jan 2010.

[29] Gregory M Cooper, Deborah A Nickerson, and Evan E Eichler. Mutational and selective effects on copy-number variants in the human genome. *Nat Genet*, 39(7 Suppl):S22–S29, Jul 2007.

[30] Richard Cordaux, Dale J Hedges, Scott W Herke, and Mark A Batzer. Estimating the retrotransposition rate of human alu elements. *Gene*, 373:134–7, May 2006.

[31] 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):154–8, Jul 2007.

[32] Graham Cormode, Mike Paterson, Süleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *SODA*, pages 197–206, 2000.

[33] P L Deininger, D J Jolly, C M Rubin, T Friedmann, and C W Schmid. Base sequence studies of 300 nucleotide renatured repeated human dna clones. *J Mol Biol*, 151(1):17–33, Sep 1981.

[34] Ian M Dew, Brian Walenz, and Granger Sutton. A tool for analyzing mate pairs in assemblies (TAMPA). *J Comput Biol*, 12(5):497–513, Jun 2005.

[35] Evan E. Eichler, Deborah A. Nickerson, David Altshuler, Anne M. Bowcock, Lisa D. Brooks, Nigel P. Carter, Deanna M. Church, Adam Felsenfeld, Mark Guyer, Charles Lee, James R. Lupski, James C. Mullikin, Jonathan K. Pritchard, Jonathan Sebat, Stephen T. Sherry, Douglas Smith, David Valle, and Robert H. Waterston. Completing the map of human genetic variation. *Nature*, 447:161–165, 10 May 2007.

[36] Evan E Eichler and Andrew W Zimmerman. A hot spot of genetic instability in autism. *N Engl J Med*, 358(7):737–9, Feb 2008.

[37] Adam D Ewing and Haig H Kazazian, Jr. High-throughput sequencing reveals extensive variation in human-specific l1 content in individual human genomes. *Genome Res*, 20(9):1262–70, Sep 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] Klaus Fellermann, Daniel E Stange, Elke Schaeffeler, Hartmut Schmalzl, Jan Wehkamp, Charles L Bevins, Walter Reinisch, Alexander Teml, Matthias Schwab, Peter Lichter, Bernhard Radlwimmer, and Eduard F Stange. A chromosome 8 gene-cluster polymorphism with low human beta-defensin 2 gene copy number predisposes to crohn disease of the colon. *Am J Hum Genet*, 79(3):439–48, Sep 2006.

[40] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *FOCS*, pages 390–398, 2000.

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

[42] Daya Ram Gaur, Ramesh Krishnamurti, and Rajeev Kohli. The capacitated max -cut problem. *Math. Program.*, 115(1):65–72, 2008.

[43] Enrique Gonzalez, Hemant Kulkarni, Hector Bolivar, Andrea Mangano, Racquel Sanchez, Gabriel Catano, Robert J Nibbs, Barry I Freedman, Marlon P Quinones, Michael J Bamshad, Krishna K Murthy, Brad H Rovin, William Bradley, Robert A Clark, Stephanie A Anderson, Robert J O'connell, Brian K Agan, Seema S Ahuja, Rosa Bologna, Luisa Sen, Matthew J Dolan, and Sunil K Ahuja. The influence of ccl3l1 gene-containing segmental duplications on hiv-1/aids susceptibility. *Science*, 307(5714):1434–40, Mar 2005.

[44] 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, Zeljko Kucan, Ivan Gusic, 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):710–22, May 2010.

[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, F. Hormozdiari, I. Birol, E. E. Eichler, and S.C. Sahinalp. Cache oblivious algorithms for high throughput read mapping. *Nature Methods*, 7, 2010.

[47] 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):1277–83, May 2010.

[48] Iman Hajirasouliha, Fereydoun Hormozdiari, S Cenk Sahinalp, and Inanc Birol. Optimal pooling for genome re-sequencing with ultra-high-throughput short-read technologies. *Bioinformatics*, 24(13):i32–40, Jul 2008.

[49] M.M. Halldrsson. Approximating the minimum maximal independence number. *Inform. Process. Lett.*, 46:169–172, 1993.

[50] Ingo Helbig, Heather C Mefford, Andrew J Sharp, Michel Guipponi, Marco Fichera, Andre Franke, Hiltrud Muhle, Carolien de Kovel, Carl Baker, Sarah von Spiczak, Katherine L Kron, Ines Steinich, Ailing A Kleefuss-Lie, Costin Leu, Verena Gaus, Bettina Schmitz, Karl M Klein, Philipp S Reif, Felix Rosenow, Yvonne Weber, Holger Lerche, Fritz Zimprich, Lydia Urak, Karoline Fuchs, Martha Feucht, Pierre Genton, Pierre Thomas, Frank Visscher, Gerrit-Jan de Haan, Rikke S Møller, Helle Hjalgrim, Daniela Luciano, Michael Wittig, Michael Nothnagel, Christian E Elger, Peter Nürnberg, Corrado Romano, Alain Malafosse, Bobby P C Koeleman, Dick Lindhout, Ulrich Stephani, Stefan Schreiber, Evan E Eichler, and Thomas Sander. 15q13.3 microdeletions increase risk of idiopathic generalized epilepsy. *Nat Genet*, 41(2):160–2, Feb 2009.

[51] LaDeana W Hillier, Gabor T Marth, Aaron R Quinlan, David Dooling, Ginger Fewell, Derek Barnett, Paul Fox, Jarret I Glasscock, Matthew Hickenbotham, Weichun Huang, Vincent J Magrini, Ryan J Richt, Sacha N Sander, Donald A Stewart, Michael Stromberg, Eric F Tsung, Todd Wylie, Tim Schedl, Richard K Wilson, and Elaine R Mardis. Whole-genome sequencing and variant discovery in C. elegans. *Nat Methods*, 5(2):183–188, Feb 2008.

[52] Emily Hodges, Zhenyu Xuan, Vivekanand Balija, Melissa Kramer, Michael N Molla, Steven W Smith, Christina M Middle, Matthew J Rodesch, Thomas J Albert, Gregory J Hannon, and W Richard McCombie. Genome-wide in situ exon capture for selective re-sequencing. *Nat Genet*, 39(12):1522–7, Dec 2007.

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

[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):1915–21, Jul 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 Res*, 21(6):840–9, Jun 2011.

[57] Fereydoun Hormozdiari, Petra Berenbrink, Natasa Przulj, and S. Cenk Sahinalp. Not all scale-free networks are born equal: The role of the seed graph in ppi network evolution. *PLoS Comput Biol*, 3(7):e118, Jul 2007.

[58] 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. *ISMB/Bioinformatics*, 26(12):i350–7, Jun 2010.

[59] Fereydoun Hormozdiari, Iman Hajirasouliha, Andrew McPherson, Evan E. Eichler, and Süleyman Cenk Sahinalp. Simultaneous structural variation discovery in multiple paired-end sequenced genomes. In *RECOMB*, pages 104–105, 2011.

[60] Fereydoun Hormozdiari, Raheleh Salari, Michael Hsing, Alexander Schönhuth, Simon K Chan, S Cenk Sahinalp, and Artem Cherkasov. The effect of insertions and deletions on wirings in protein-protein interaction networks: a large-scale study. *J Comput Biol*, 16(2):159–67, Feb 2009.

[61] C M Houck, F P Rinehart, and C W Schmid. A ubiquitous family of repeated dna sequences in the human genome. *J Mol Biol*, 132(3):289–306, Aug 1979.

[62] 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):1171–82, Jun 2010.

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

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

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

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

[67] International Cancer Genome Consortium. International network of cancer genome projects. *Nature*, 464(7291):993–8, Apr 2010.

[68] 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):1253–61, Jun 2010.

[69] Neil C. Jones and Pavel Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT press, 2004.

[70] J. Jurka, V. V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase update, a database of eukaryotic repetitive elements. *Cytogenet Genome Res*, 110(1-4):462–467, 2005.

[71] 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.

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

[73] 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 Tzn, 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.

[74] 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):837–47, Nov 2010.

[75] 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 Methods*, 7(5):365–71, May 2010.

[76] 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):1011–5, Aug 2009.

[77] 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 Biotechnol*, 29(1):59–63, Jan 2011.

[78] 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.

[79] 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.

[80] 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 Biol*, 8(10):R224, 2007.

[81] Hugo Y K Lam, Xinmeng Jasmine Mu, Adrian M Stütz, Andrea Tanzer, Philip D Cayting, Michael Snyder, Philip M Kim, Jan O Korbel, and Mark B Gerstein. Nucleotide-resolution analysis of structural variants using breakseq and a breakpoint library. *Nat Biotechnol*, 28(1):47–55, Jan 2010.

[82] E S Lander, L M Linton, B Birren, C Nusbaum, M C Zody, J Baldwin, K Devon, K Dewar, M Doyle, W FitzHugh, R Funke, D Gage, K Harris, A Heaford, J Howland, L Kann, J Lehoczky, R LeVine, P McEwan, K McKernan, J Meldrim, J P Mesirov, C Miranda, W Morris, J Naylor, C Raymond, M Rosetti, R Santos, A Sheridan, C Sougnez, N Stange-Thomann, N Stojanovic, A Subramanian, D Wyman, J Rogers, J Sulston, R Ainscough, S Beck, D Bentley, J Burton, C Clee, N Carter, A Coulson, R Deadman, P Deloukas, A Dunham, I Dunham, R Durbin, L French, D Grafham, S Gregory, T Hubbard, S Humphray, A Hunt, M Jones, C Lloyd, A McMurray, L Matthews, S Mercer, S Milne, J C Mullikin, A Mungall, R Plumb, M Ross, R Shownkeen, S Sims, R H Waterston, R K Wilson, L W Hillier, J D McPherson,

M A Marra, E R Mardis, L A Fulton, A T Chinwalla, K H Pepin, W R Gish, S L Chissoe, M C Wendl, K D Delehaunty, T L Miner, A Delehaunty, J B Kramer, L L Cook, R S Fulton, D L Johnson, P J Minx, S W Clifton, T Hawkins, E Branscomb, P Predki, P Richardson, S Wenning, T Slezak, N Doggett, J F Cheng, A Olsen, S Lucas, C Elkin, E Uberbacher, M Frazier, R A Gibbs, D M Muzny, S E Scherer, J B Bouck, E J Sodergren, K C Worley, C M Rives, J H Gorrell, M L Metzker, S L Naylor, R S Kucherlapati, D L Nelson, G M Weinstock, Y Sakaki, A Fujiyama, M Hattori, T Yada, A Toyoda, T Itoh, C Kawagoe, H Watanabe, Y Totoki, T Taylor, J Weissenbach, R Heilig, W Saurin, F Artiguenave, P Brottier, T Bruls, E Pelletier, C Robert, P Wincker, D R Smith, L Doucette-Stamm, M Rubenfield, K Weinstock, H M Lee, J Dubois, A Rosenthal, M Platzer, G Nyakatura, S Taudien, A Rump, H Yang, J Yu, J Wang, G Huang, J Gu, L Hood, L Rowen, A Madan, S Qin, R W Davis, N A Federspiel, A P Abola, M J Proctor, R M Myers, J Schmutz, M Dickson, J Grimwood, D R Cox, M V Olson, R Kaul, C Raymond, N Shimizu, K Kawasaki, S Minoshima, G A Evans, M Athanasiou, R Schultz, B A Roe, F Chen, H Pan, J Ramser, H Lehrach, R Reinhardt, W R McCombie, M de la Bastide, N Dedhia, H Blöcker, K Hornischer, G Nordsiek, R Agarwala, L Aravind, J A Bailey, A Bateman, S Batzoglou, E Birney, P Bork, D G Brown, C B Burge, L Cerutti, H C Chen, D Church, M Clamp, R R Copley, T Doerks, S R Eddy, E E Eichler, T S Furey, J Galagan, J G Gilbert, C Harmon, Y Hayashizaki, D Haussler, H Hermjakob, K Hokamp, W Jang, L S Johnson, T A Jones, S Kasif, A Kaspryzk, S Kennedy, W J Kent, P Kitts, E V Koonin, I Korf, D Kulp, D Lancet, T M Lowe, A McLysaght, T Mikkelsen, J V Moran, N Mulder, V J Pollara, C P Ponting, G Schuler, J Schultz, G Slater, A F Smit, E Stupka, J Szustakowski, D Thierry-Mieg, J Thierry-Mieg, L Wagner, J Wallis, R Wheeler, A Williams, Y I Wolf, K H Wolfe, S P Yang, R F Yeh, F Collins, M S Guyer, J Peterson, A Felsenfeld, K A Wetterstrand, A Patrinos, M J Morgan, P de Jong, J J Catanese, K Osoegawa, H Shizuya, S Choi, Y J Chen, J Szustakowki, and International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, Feb 2001.

[83] Rebecca J Leary, Isaac Kinde, Frank Diehl, Kerstin Schmidt, Chris Clouser, Cisilya Duncan, Alena Antipova, Clarence Lee, Kevin McKernan, Francisco M De La Vega, Kenneth W Kinzler, Bert Vogelstein, Luis A Diaz, Jr, and Victor E Velculescu. Development of personalized tumor biomarkers using massively parallel sequencing. *Sci Transl Med*, 2(20):20ra14, Feb 2010.

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

[85] 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.

[86] Seunghak Lee, Eric Xing, and Michael Brudno. Mogul: Detecting common insertions and deletions in a population. In *RECOMB*, pages 357–368, 2010.

[87] 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.

[88] 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.

[89] Ruiqiang Li, Yingrui Li, Hancheng Zheng, Ruibang Luo, Hongmei Zhu, Qibin Li, Wubin Qian, Yuanyuan Ren, Geng Tian, Jinxiang Li, Guangyu Zhou, Xuan Zhu, Honglong Wu, Junjie Qin, Xin Jin, Dongfang Li, Hongzhi Cao, Xueda Hu, Hélène Blanche, Howard Cann, Xiuqing Zhang, Songgang Li, Lars Bolund, Karsten Kristiansen, Huanming Yang, Jun Wang, and Jian Wang. Building the sequence map of the human pan-genome. *Nat Biotechnol*, 28(1):57–63, Jan 2010.

[90] 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 Res*, 20(2):265–72, Feb 2010.

[91] Yingrui Li, Yujie Hu, Lars Bolund, and Jun Wang. State of the art de novo assembly of human genomes from massively parallel sequencing data. *Human Genomics*, 4(4):1473–9542, 2010.

[92] George E Liu, Can Alkan, Lu Jiang, Shaying Zhao, and Evan E Eichler. Comparative analysis of alu repeats in primate genomes. *Genome Res*, 19(5):876–85, May 2009.

[93] Elaine R Mardis. The impact of next-generation sequencing technology on genetics. *Trends Genet*, 24(3):133–141, Mar 2008.

[94] Marcel Margulies, Michael Egholm, William E Altman, Said Attiya, Joel S Bader, Lisa A Bemben, Jan Berka, Michael S Braverman, Yi-Ju Chen, Zhoutao Chen, Scott B Dewell, Lei Du, Joseph M Fierro, Xavier V Gomes, Brian C Godwin, Wen He, Scott Helgesen, Chun Heen Ho, Chun He Ho, Gerard P Irzyk, Szilveszter C Jando, Maria L I Alenquer, Thomas P Jarvie, Kshama B Jirage, Jong-Bum Kim, James R Knight, Janna R Lanza, John H Leamon, Steven M Lefkowitz, Ming Lei, Jing Li, Kenton L Lohman, Hong Lu, Vinod B Makhijani, Keith E McDade, Michael P McKenna, Eugene W Myers, Elizabeth Nickerson, John R Nobile, Ramona Plant, Bernard P Puc, Michael T Ronan, George T Roth, Gary J Sarkis, Jan Fredrik Simons, John W Simpson, Maithreyan Srinivasan, Karrie R Tartaro, Alexander Tomasz, Kari A Vogt, Greg A Volkmer, Shally H Wang, Yong Wang, Michael P Weiner, Pengguang Yu, Richard F Begley, and Jonathan M Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–80, Sep 2005.

[95] Tomas Marques-Bonet, Jeffrey M Kidd, Mario Ventura, Tina A Graves, Ze Cheng, LaDeana W Hillier, Zhaoshi Jiang, Carl Baker, Ray Malfavon-Borja, Lucinda A Fulton, Can Alkan, Gozde Aksay, Santhosh Girirajan, Priscillia Siswara, Lin Chen, Maria Francesca Cardone, Arcadi Navarro, Elaine R Mardis, Richard K Wilson, and Evan E Eichler. A burst of segmental duplications in the genome of the african great ape ancestor. *Nature*, 457(7231):877–81, Feb 2009.

[96] 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.

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

[98] 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 Res*, 19(9):1527–41, Sep 2009.

[99] Andrew McPherson, Fereydoun Hormozdiari, Abdalnasser Zayed, Ryan Giuliany, Gavin Ha, Mark G F 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 Comput Biol*, 7(5):e1001138, May 2011.

[100] 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):1481–8, Jun 2011.

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

[102] Paul Medvedev, Marc Fiume, Misko Dzamba, Tim Smith, and Michael Brudno. Detecting copy number variation with mated short reads. *Genome Res*, 20(11):1613–22, Nov 2010.

[103] Paul Medvedev, Konstantinos Georgiou, Gene Myers, and Michael Brudno. Computability of models for sequence assembly. In *WABI*, pages 289–301, 2007.

[104] Heather C Mefford, Severine Clauin, Andrew J Sharp, Rikke S Moller, Reinhard Ullmann, Raj Kapur, Dan Pinkel, Gregory M Cooper, Mario Ventura, H Hilger Ropers, Niels Tommerup, Evan E Eichler, and Christine Bellanne-Chantelot. Recurrent reciprocal genomic rearrangements of 17q12 are associated with renal disease, diabetes, and epilepsy. *Am J Hum Genet*, 81(5):1057–69, Nov 2007.

[105] 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.

[106] Ryan E Mills, Christopher T Luttig, Christine E Larkins, Adam Beauchamp, Circe Tsui, W. Stephen Pittard, and Scott E Devine. An initial map of insertion and deletion (INDEL) variation in the human genome. *Genome Res*, 16(9):1182–1190, Sep 2006.

[107] Ryan E Mills, Klaudia Walter, Chip Stewart, Robert E Handsaker, Ken Chen, Can Alkan, Alexej Abyzov, Seungtai Chris Yoon, Kai Ye, R Keira Cheetham, Asif Chinwalla, Donald F Conrad, Yutao Fu, Fabian Grubert, Iman Hajirasouliha, Fereydoun Hormozdiari, Lilia M Iakoucheva, Zamin Iqbal, Shuli Kang, Jeffrey M Kidd, Miriam K Konkel, Joshua Korn, Ekta Khurana, Deniz Kural, Hugo Y K Lam, Jing Leng, Ruiqiang Li, Yingrui Li, Chang-Yun Lin, Ruibang Luo, Xinmeng Jasmine Mu, James Nemesh, Heather E Peckham, Tobias Rausch, Aylwyn Scally, Xinghua Shi, Michael P Stromberg, Adrian M Stütz, Alexander Eckehart Urban, Jerilyn A Walker, Jiantao Wu, Yujun Zhang, Zhengdong D Zhang, Mark A Batzer, Li Ding, Gabor T Marth, Gil McVean, Jonathan Sebat, Michael Snyder, Jun Wang, Kenny Ye, Evan E Eichler, Mark B Gerstein, Matthew E Hurles, Charles Lee, Steven A McCarroll, Jan O Korbel, and 1000 Genomes Project. Mapping copy number variation by population-scale genome sequencing. *Nature*, 470(7332):59–65, Feb 2011.

[108] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[109] S. Muthukrishnan and Süleyman Cenk Sahinalp. Approximate nearest neighbors and sequence comparison with block operations. In *STOC*, pages 416–424, 2000.

[110] Eugene W. Myers. The fragment assembly string graph. In *ECCB/JBI*, page 85, 2005.

[111] Z. Ning, A. J. Cox, and J. C. Mullikin. SSAHA: a fast search method for large DNA databases. *Genome Res*, 11(10):1725–1729, Oct 2001.

[112] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes (extended abstract). pages 229–234, 1988.

[113] 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):400–5, May 2010.

[114] 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.

[115] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. A new approach to fragment assembly in dna sequencing. In *RECOMB*, pages 256–267, 2001.

[116] Mihai Pop and Steven L Salzberg. Bioinformatics challenges of new sequencing technology. *Trends Genet*, 24(3):142–149, Mar 2008.

[117] Alkes L Price, Eleazar Eskin, and Pavel A Pevzner. Whole-genome analysis of alu repeat elements reveals complex evolutionary history. *Genome Res*, 14(11):2245–52, Nov 2004.

[118] 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.

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

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

[121] Richard Redon, Shumpei Ishikawa, Karen R Fitch, Lars Feuk, George H Perry, T. Daniel Andrews, Heike Fiegler, Michael H Shapero, Andrew R Carson, Wenwei Chen, Eun Kyung Cho, Stephanie Dallaire, Jennifer L Freeman, Juan R Gonzlez, Mnica Gratacs, Jing Huang, Dimitrios Kalaitzopoulos, Daisuke Komura, Jeffrey R MacDonald, Christian R Marshall, Rui Mei, Lyndal Montgomery, Kunihiro Nishimura, Kohji Okamura, Fan Shen, Martin J Somerville, Joelle Tchinda, Armand Valsesia, Cara Woodwark, Fengtang Yang, Junjun Zhang, Tatiana Zerjal, Jane Zhang, Lluis Armengol, Donald F Conrad, Xavier Estivill, Chris Tyler-Smith, Nigel P Carter, Hiroyuki Aburatani, Charles Lee, Keith W Jones, Stephen W Scherer, and Matthew E Hurles. Global variation in copy number in the human genome. *Nature*, 444(7118):444–454, Nov 2006.

[122] Anna M. Ritz, Ali Bashir, and Benjamin J. Raphael. Structural variation analysis with strobe reads. *Bioinformatics*, 26(10):1291–1298, 2010.

[123] J C Roach, C Boysen, K Wang, and L Hood. Pairwise end sequencing: a unified approach to genomic mapping and sequencing. *Genomics*, 26(2):345–53, Mar 1995.

[124] Süleyman Cenk Sahinalp. Edit distance under block operations. In *Encyclopedia of Algorithms*. 2008.

[125] Süleyman Cenk Sahinalp and Uzi Vishkin. Symmetry breaking for suffix tree construction. In *STOC*, pages 300–309, 1994.

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

[127] Abdel-Halim Salem, Gail E Kilroy, W Scott Watkins, Lynn B Jorde, and Mark A Batzer. Recently integrated alu elements and human genomic diversity. *Mol Biol Evol*, 20(8):1349–61, Aug 2003.

[128] F Sanger, A R Coulson, B G Barrell, A J Smith, and B A Roe. Cloning in single-stranded bacteriophage as an aid to rapid dna sequencing. *J Mol Biol*, 143(2):161–78, Oct 1980.

[129] F Sanger, A R Coulson, G F Hong, D F Hill, and G B Petersen. Nucleotide sequence of bacteriophage lambda dna. *J Mol Biol*, 162(4):729–73, Dec 1982.

[130] F Sanger, S Nicklen, and A R Coulson. Dna sequencing with chain-terminating inhibitors. 1977. *Biotechnology*, 24:104–8, 1992.

[131] C W Schmid and P L Deininger. Sequence organization of the human genome. *Cell*, 6(3):345–58, Nov 1975.

[132] 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):943–7, Feb 2010.

[133] J. Sebat, B. Lakshmi, J. Troge, J. Alexander, J. Young, P. Lundin, S. Maner, H. Massa, M. Walker, M. Chi, N. Navin, R. Lucito, J. Healy, J. Hicks, K. Ye, A. Reiner, T. C. Gilliam, B. Trask, N. Patterson, A. Zetterberg, and M. Wigler. Large-scale copy number polymorphism in the human genome. *Science*, 305(5683):525–8, 2004.

[134] Andrew J Sharp, Ze Cheng, and Evan E Eichler. Structural variation of the human genome. *Annu Rev Genomics Hum Genet*, 7:407–442, 2006.

[135] Jay Shendure, Robi D Mitra, Chris Varma, and George M Church. Advanced sequencing technologies: methods and goals. *Nat Rev Genet*, 5(5):335–344, May 2004.

[136] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven J M Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome Res*, 19(6):1117–23, Jun 2009.

[137] 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.

[138] Paweł Stankiewicz and James R Lupski. Structural variation in the human genome and its role in disease. *Annu Rev Med*, 61:437–55, 2010.

[139] Chip Stewart and et al. Spanner. *Unpublished*.

[140] 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*, 330(6004):641–6, Oct 2010.

[141] 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):e484, 2007.

[142] M. Tang, M. Waterman, and S. Yooseph. Zinc finger gene clusters and tandem gene duplication. *J. Comput. Biol.*, 9(2):429–446, 2002.

[143] Emily H Turner, Sarah B Ng, Deborah A Nickerson, and Jay Shendure. Methods for genomic partitioning. *Annu Rev Genomics Hum Genet*, 10:263–84, 2009.

[144] 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.

[145] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[146] Mario Ventura, Claudia R Catacchio, Can Alkan, Tomas Marques-Bonet, Saba Sajjadian, Tina A Graves, Fereydoun Hormozdiari, Arcadi Navarro, Maika Malig, Carl Baker, Choli Lee, Emily H Turner, Lin Chen, Jeffrey M Kidd, Nicoletta Archidiacono, Jay Shendure, Richard K Wilson, and Evan E Eichler. Gorilla genome structural variation reveals evolutionary parallelisms with chimpanzee. *Genome Res*, Jun 2011.

[147] 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.

[148] Menno R Vriens, Jennifer Mj Schreinemakers, Insoo Suh, Marlon A Guerrero, and Orlo H Clark. Diagnostic markers and prognostic factors in thyroid cancer. *Future Oncol*, 5(8):1283–93, Oct 2009.

[149] Jianxin Wang, Lei Song, Deepak Grover, Sami Azrak, Mark A Batzer, and Ping Liang. dbrip: a highly integrated database of retrotransposon insertion polymorphisms in humans. *Hum Mutat*, 27(4):323–9, Apr 2006.

[150] 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.

[151] 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):500–1, Feb 2007.

[152] 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.

[153] David J Witherspoon, Jinchuan Xing, Yuhua Zhang, W Scott Watkins, Mark A Batzer, and Lynn B Jorde. Mobile element scanning (me-scan) by targeted high-throughput sequencing. *BMC Genomics*, 11:410, 2010.

[154] 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.

[155] V. Yanovsky, S.R. Rumble, and M. Brudno. Read mapping algorithms for single molecule sequencing data. In *Proceedings of Workshop on Algorithms in Bioinformatics (WABI)*, 2008.

[156] 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):2865–71, Nov 2009.

[157] Seungtai Yoon, Zhenyu Xuan, Vladimir Makarov, Kenny Ye, and Jonathan Sebat. Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res*, 19(9):1586–92, Sep 2009.

[158] Bruno Zeitouni, Valentina Boeva, Isabelle Janoueix-Lerosey, Sophie Loeillet, Patricia Legoix-né, Alain Nicolas, Olivier Delattre, and Emmanuel Barillot. Svdetect: a tool to identify genomic structural variations from paired-end and mate-pair sequencing data. *Bioinformatics*, 26(15):1895–6, Aug 2010.

[159] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Res*, 18(5):821–9, May 2008.