DEVELOPING NEW APPROACHES FOR MULTI-PLATFORM AND MULTI-INDIVIDUAL GENOMIC SEQUENCE ASSEMBLY

by

Pınar Kavak

B.S., Computer Engineering, Bilkent University, 2006

M.S., Computer Engineering, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2017

ACKNOWLEDGEMENTS

I want to give my sincere thanks to my advisors Prof. Tunga Güngör and Prof. Can Alkan for the continuous support of my Ph.D. study, as well, for their knowledge, guidance, patience and motivation. Their support made this research and thesis real. I want to thank to Prof. Can Özturan, and Prof. Arzucan Özgür, for their kindness of evaluating my progress each semester and contributing to my research with valuable suggestions in addition to being my thesis committee members, reading and commenting on my thesis. I also want to thank to Prof. Uğur Sezerman and Prof. Emre Karakoç for being my thesis committee members, reading and giving insightful comments and suggestions on my thesis. I want to thank Prof. S. Cenk Şahinalp who provided me the opportunity to join SFU Lab for Computational Biology and all lab members: Ehsan Haghshenas, Alex Gawronski, Ermin Hodžić, Salem Malikić, Mike Ford, and Hossein Asghari for the friendly lab environment. I would especially like to thank to Dr. Faraz Hach for the knowledge and support and Yen-Yi Lin for his kind help, sharing his knowledge and working with me. I want to thank to Ibrahim Numanagić for his valuable help too. I want to thank to my colleges at TÜBİTAK-İGBAM: Mete Akgün, Mahmut Ş. Sağıroğlu, Huseyin Demirci, Yıldız Uludağ, Oğuzhan Külekçi, Ahmet Çakmak, M. Yağmur Gök, Serkan Barut, Bayram Yüksel, Bekir Ergüner, Zeliha Görmez and Buğra Özer. I would like to thank Turkish Human Genome Project (TGP) members for sharing the DNA sample and data. I want to thank to the Republic of Turkey Ministry of Development Infrastructure Grant (no: 2011K120020) and BİLGEM - TÜBİTAK (The Scientific and Technological Research Council of Turkey) (grant no: T439000) for their support on the first and second studies. Special thanks go to TÜBİTAK BİDEB 2214-A programme for supporting me to visit SFU Lab for Computational Biology. I want to thank to all my friends and my supportive roommate Isla Robertson, for their spiritual support. Lastly, I would like to thank to my family: my father Arslan Kavak, my mother Gülseren Kavak, my sister Tülin Yaşar, and my brother Taşkın Kavak for their spiritual support.

ABSTRACT

DEVELOPING NEW APPROACHES FOR MULTI-PLATFORM AND MULTI-INDIVIDUAL GENOMIC SEQUENCE ASSEMBLY

High throughput sequencing (HTS) technologies generate huge amount of data with very low cost, which prompted research on algorithm development to analyze large DNA sequence datasets. In this thesis, we propose new solutions to three related problems in genomics field. Firstly, although the accuracy and reproducibility of HTS based analyses is highly improved, the usability of these platforms in terms of robustness is still an open question. We produced whole genome shotgun (WGS) sequence data from the genomes of two individuals in two different centers to assess the usability of a widely used HTS platform in terms of robustness in clinical applications. We observe that HTS platforms are powerful enough for providing data for first-pass clinical tests, but before using in clinical applications, the variant predictions need to be confirmed by orthogonal methods. Secondly, we still need innovative methods for the *de novo* genome assembly problem. The task of assembling very short DNA sequence reads into -ideally- complete chromosome sequences is further complicated due to (i) the repetitive and duplicated structure of genomes, and (ii) the fact that the data produced by the HTS technologies tend to be short and error prone. We present a new method to increase the assembly accuracy by integrating data from Illumina, Ion-Torrent and Roche-454 platforms. Lastly, characterization of novel sequence insertions longer than the average read length remains a challenging task. There are only a few algorithms that are specifically developed for novel sequence insertion discovery that can bypass the need for the whole genome *de novo* assembly. We present a new algorithm, Pamir, to efficiently and accurately discover and genotype novel sequence insertions using either single or multiple genome sequencing datasets.

ÖZET

ÇOKLU PLATFORM VE ÇOKLU BİREYDEN ELDE EDİLEN VERİLER İLE YENİ GENOM BİRLEŞTİRME YAKLAŞIMLARI GELİŞTİRME

Yüksek hacimli dizileme (YHD) teknolojileri büyük ölçüde veriyi düşük maliyete üretiyor, bu durum büyük miktardaki DNA dizisi verisini analiz etmek için algoritma geliştirme araştırmasını hızlandırdı. Bu tezde, genomiks alanında üç bağlantılı probleme yeni çözümler getiriyoruz. İlk olarak, her ne kadar YHD'ye dayalı analizlerin doğruluğu ve tekrarlanabilirliği üzerinde önemli gelişmeler kaydedilse de YHD platformlarının dayanıklılık açısından kullanılabilirliği hala açık soru. Yaygın bir YHD platformunun klinik uygulamalarda kullanılabilirliğini dayanıklılık açısından incelemek için iki bireyin genomlarının tüm genom dizileme verisini iki ayrı merkezde diziledik. YHD platformları ilk-aşama klinik tesler için veri sağlamada çok güçlüler ancak varyant tahminlerinin klinik uygulamalarda kullanılmadan önce ortogonal yöntemlerle doğrulanması gerektiği sonucunu gözlemledik. İkinci olarak genom birleştirme problemi için hala yaratıcı çözümlere ihtiyacımız var. Çok kısa DNA dizilerini -idealde- tüm kromozom dizilerine birleştirmek şu sebeplerden karmaşık bir iş (i) genomların tekrarlı ve kopyalı yapısı (ii) YHD'nin ürettiği verinin kısa ve hatalı olması. Birleştirme doğruluğunu artırmak için üç farklı teknolojiden (Illumina, Ion-Torrent, Roche-454) veri dahil eden yeni bir metod sunuyoruz. Son olarak, ortalama dizi parçacığından uzun yeni dizi insersiyonlarının tanımlanması hala zorlu bir iş. Özellikle yeni dizi insersiyon bulma için geliştirilmiş ve yeniden tüm genom birleştirmesini es geçebilecek az sayıda algoritma var. Tek veya çok genom dizisi verisetlerinde verimli ve doğru bir şekilde yeni dizi insersiyonlarını bulan ve genotipleme yapan yeni bir algoritma Pamir'i sunuyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$3'$	The "tail" end of the DNA strand that has the hidroxyl group at the end of the third carbon in the sugar ring
$5'$	The end of the DNA strand that has the fifth carbon in the sugar ring of the deoxyribose at the end
$<_v$	Ordering of vertices
a	Leftmost mapping locus of an OEA
B_1	SNP/Indels from pooled data genotyped within the BGI (1st individual)
$f(v)$	Maximum weight from the root to any vertex v
E	Total number of edges in graph G
e_{mv}	Edge connecting vertices m and v
G	Graph
i_l	Number of reads that align across the left breakpoint location in I
i_r	Number of reads that align across the right breakpoint location in I
I	Temporary insertion sequence: Lbp upstream of the breakpoint from the reference plus the insertion plus Lbp downstream from the reference
L	Fragment length
m	Second vertex
p	Insertion breakpoint location
r	Root of the graph
re	Number of reads that align across the breakpoint location in RE
R	Number of reads in the given cluster
RE	Temporary reference sequence: L bp upstream of the breakpoint from the reference plus L bp downstream of the breakpoint from the reference
S_1	Sample of the 1st individual

S_2	Sample of the 2nd individual
S_{1B}	The data generated at BGI (1st individual)
S_{2B}	The data generated at BGI (2nd individual)
S_{1BT}	Pooled callset of the 1st individual
S_{2BT}	Pooled callset of the 2nd individual
S_{1T}	The data generated at TÜBİTAK (1st individual)
S_{2T}	The data generated at TÜBİTAK (2nd individual)
T_1	SNP/Indels genotyped within the TÜBİTAK (1st individual)
T_2	SNP/Indels genotyped within the TÜBİTAK (2nd individual)
v	Vertex
w_{mv}	Maximum prefix-suffix overlap between the reads represented by m and v
x	Ratio between i and re
γ	Threshold value for genotyping

LIST OF ACRONYMS/ABBREVIATIONS

ABI SOLID	Applied Biosystems Sequencing by Oligonucleotide Ligation and Detection
ABySS	Assembly By Short Sequences
ANNOVAR	functional ANNOtation of genetic VARiations
BAC	Bacterial Artificial Chromosome
BAM	Binary format for sAM
BLAST	Basic Local Alignment Search Tool
BWA	Burrows-Wheeler Aligner
BGI	Beijing Genomics Institute
CABOG	Celera Assembler with the Best Overlap Graph
CNV	Copy Number Variation
dbSNP	The Single Nucleotide Polymorphism database
DNA	DeoxyriboNucleic Acid
E.coli	Escherichia coli
EST	Expressed Sequence Tag
ERV	Endogenous retroviruses
FastQC	Fastq Quality Control
FN	False Negative
FP	False Positive
GATK	Genome Analysis Toolkit
GRCh	Genome Reference Consortium human genome
HC	Homopolymer Compression
HTS	High Throughput Sequencing
İGBAM	İleri Genom ve Biyoenformatik Araştırma Merkezi/Advanced Genomics and Bioinformatics Research Center
İNAREK	İnsan Araştırmaları Kurumsal Değerlendirme Kurulu/ Committee on Ethical Conduct in Studies Involving Human Subjects
LINE	Long Interspersed Nuclear Elements

LRC	Long Read Contig
LTR	Long Terminal Repeat
NCBI	National Center for Biotechnology Information
ncRNA	non-coding RibuNucleic Acid
NGS	Next Generation Sequencing
OEA	One End Anchor
OLC	Overlap Layout Consensus
NP-hard	Non-deterministic Polynomial time hard
PacBio	Pacific Biosciences
PbCR	Pacbio Corrected Reads
PCR	Polymerase Chain Reaction
PopIns	Population scale detection of novel sequence Insertions
SAM	Sequence Alignment/Map
SGA	String Graph Assembler
SINE	Short Interspersed Nuclear Elements
SMRT-SV	Single Molecule Real Time - Structural Variant
SNP	Single Nucleotide Polymorphism
SPAdes	St. Petersburg genome Assembler
SRC	Short Read Contig
SRA	Sequence Read Archive
SV	Structural Variation
TP	True Positive
TÜBİTAK	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu/ The Scientific and Technological Research Council of Turkey
UCSC	University of California Santa Cruz
UTR3	$3^{'}$ UnTranslated Region
UTR5	$5^{'}$ UnTranslated Region
VCF	Variant Call Format
VQSR	Variant Quality Score Recalibration
WGS	Whole Genome Shotgun
YHD	Yüksek Hacimde Dizileme

1. INTRODUCTION

Next generation sequencing (NGS) technologies have provided a great opportunity to the researchers to investigate whole human and other species genomes. After traditional sequencing methods such as Maxam-Gilbert [5] and Sanger sequencing [6] revolution in sequencing in 1970s, it took researchers almost 30 years to develop new approaches to enable faster and cheaper sequencing. Array based technologies in which DNA fragments of specific chromosomal loci are fixed on a surface with either bacterial artificial chromosome clones or oligonucleotide synthesis were considered in clinical applications first, but they have limitations such as non-uniform coverage, low accuracy on low coverage regions, missing aneuploidies and marker chromosomes [7]. They also need high amount of time (more than two days or longer) and financial requirements compared to high throughput sequencing (HTS) technologies. On the other hand, HTS technologies have many advantages over array based techniques and resolve most of their limitations. Sequencing by synthesis, sequencing by ligation, and single molecule sequencing are the general titles of these new methods. Sequencing by synthesis takes a single strand of DNA and synthesizes its complementary strand enzymatically, one base pair at a time. Sequencing by ligation uses DNA ligase enzyme to identify nucleotides at the given positions in a DNA sequence. In single molecule sequencing, DNA fragment is added with poly-A tail adaptors which are attached to the flow cell surface and extension based sequencing is applied.

Currently, the most widely used platforms use sequencing by synthesis technology, each of which has many advantages and/or disadvantages over the other according to the purposes and financial issues. Illumina, Roche (454), PacBio, Ion-Torrent are the most popular sequencing by synthesis platforms in use. The common characteristic of next generation data generated by these platforms is that the reads are short (except Pacbio), i.e. between 100bp (Illumina/Ion-Torrent) - 400bp (454). In order to make an assembly with short reads, high coverage data are needed and this increases complexity, also repeat resolving becomes a big problem with short reads [8]. Pacific Biosciences platform generates longer reads than other platforms, but it has lower accuracy, i.e.

>= %14 error rate. The data generated by each of these platforms have their own characteristics, benefits and deficits; e.g. none of them alone is appropriate for generating a perfect assembly. With the development of these new sequencing technologies, the cost of obtaining whole genome sequences has decreased in the following years which also led to an increase in the amount of the produced data. The generation of the huge amount of data by high throughput sequencing (HTS) platforms also increased the amount of research to investigate them and the number of researchers interested in exploring the data. In this thesis, we concentrate on three separate problems that are related to next generation sequencing technologies in genomics.

First, this increase made more clinical sequencing projects such as Genomics England and ClinSeq applicable. High throughput sequencing (HTS) based analysis is improved significantly in terms of accuracy and reproducibility, but the usability of these types of data for diagnostic and prognostic applications requires a near perfect data generation. Although the performance of HTS platforms has been tested in various studies [9–11], the robustness of HTS platforms still needs to be systematically assessed. As our first study, we perform an experiment to analyze the robustness of HTS platforms.

Second, the two main problems of analyzing HTS data are the sequence alignment and the *de novo* assembly. Although including its own existing problems, sequence alignment became simpler and more useful than before, and gave chance to study identifying variations which lead to genetic diseases. Complex disease problems also have a wide range of data to be worked on, but their identification is harder than identifying Mendelian disorders. After the draft human reference genome was produced in 2001 [12], genome alignment has got importance to analyze new individuals for diseases and population genetic studies, but genome assembly problem did not get much attention. There are still many species which are waiting for their reference genomes to be sequenced or constructed. Human genome has also some unknown sections because of the repetitive structure of the genome itself. In addition, it is generated by a limited number of human DNAs which cannot represent all populations because of the missing sequences through the human lineage. These unknown genomic

regions or whole genomes still wait for our exploration. Therefore, the second problem we scrutinize is improving the *de novo* genome assemblies of the whole genomes.

Genome assembly is a very important problem because of the necessity of the discovery of these unknown genomes and genomic regions. There are various solutions to the genome assembly problem which are effectively in use, but the problem still has very important issues to be solved. The accuracy of the results of the existing assemblers is not as high as they should be which makes it still remain as an open problem. The source of the assembly problem comes first from the nature of the genome structure. The genome has a repetitive structure and next generation sequencing technologies are not able to sequence long-enough reads that will cover these repeats. Although some of them (Pacific Biosciences) are able to sequence long sequences, the error-rate is too high (>= 30%). There are three kinds of assemblers introduced until now. Greedy assemblers use simple mappings and work on a greedy basis. They are incapable of assembling huge data. Overlap layout consensus (OLC) graph based assemblers use pairwise alignments to find a layout and generate a consensus assembly; they usually work well with long reads. De-Bruijn graph based assemblers use a k-mer graph approach, and they are successful with the short reads. OLC graph based assemblers and de-Bruijn graph assemblers have their own advantages on specific cases, but none of them is a state of the art assembler yet. The problem of these assemblers is that they are usually developed for specific kind of data, so they might not work as well with another kind of data. There are also some methods which use different types of data, called hybrid assemblers.

The last problem we explore is discovering novel sequence insertions on whole genome sequence data. Genomic structural variations (SVs) which affect more than 50 base pairs (bp) can be deletions, insertions, inversions, duplications and retrotranspositions [13, 14]. The characterization of insertions is important because they might also exist in coding regions [15]. However, insertions have not been worked on as much as other structural variations mainly because of the lack of long read sequences, and it needs sequence assembly which is a computationally challenging task. The studies on discovering the novel sequence insertions need to be improved. The third problem we

will examine is novel sequence insertion discovery.

Within this context, we present three different yet complementary studies introducing new approaches to the three problems of the genomics field. As the first study, we analyze the reliability of a largely used HTS platform, i.e. Illumina, in terms of reproducibility and also some of the most popular data analysis tools with an experiment. In this experiment, we sequenced the DNAs of two donors twice with the same machine and analyzed the sequencing data with the same tools. In the second study, we propose a hybrid multi-platform sequence data approach to improve genome assemblies. Finally, we present a new algorithm for novel insertion discovery and our tool Pamir which we developed for the discovery and genotyping of novel sequence insertions in the genomes of one or more individuals. We explain each problem in detail in the regarding sections.

1.1. Contributions

1.1.1. Reproducibility of NGS

In the first study, we aim to investigate the usability of a widely used HTS platform, i.e. Illumina, in terms of robustness, for usage in accurate and reproducible clinical applications. For this purpose, we generated whole genome shotgun (WGS) sequence data from the genomes of two human individuals with the same model of Illumina platform in two different genome sequencing centers. We used the same mapping and analysis tools (BWA, SAMtools and GATK) to characterize SNPs and Indels. After analyzing the data, we observed significant number of discrepancies in the call sets. Although most of the disagreements between the call sets were found within genomic regions containing common repeats and segmental duplications as expected, only a small fraction of the discordant variants were within the exons and other functionally relevant regions such as promoters. After our analysis, we came to a conclusion that the variant predictions still need to be validated using orthogonal methods [16] before using them in clinical applications even though HTS platforms are very powerful for providing data for first-pass clinical tests.

1.1.2. Improving genome assemblies

In this study, our purpose is to propose a new approach to the *de novo* assembly problem using multi-platform sequence data to increase the quality of the final assembly with combination of short reads and long reads that belong to the same sample. Short reads are from Illumina and long reads are from both 454 and Ion-Torrent. We assemble different types of reads separately with appropriate assemblers. We obtain the first group of contigs of the short reads with a de-Bruijn graph based assembler. In a de-Bruijn graph based assembler nodes represent the k-mers ($<$ read length) chopped from the reads and edges represent the k-1 length overlaps between the k-mers on the nodes. In this assembly algorithm, one follows an Eulerian path, i.e. each edge is visited exactly once, on the graph to discover a consensus sequence. We assemble the long reads into long-read contigs with OLC graph based assemblers where each node represents a read and each edge represents the overlaps between the reads. After graph generation, a Hamiltonian path (i.e. each node is visited exactly once) is followed to determine the consensus sequence. After finding the consensus sequences with both short reads and long reads, we use the short-read contigs to correct the long read contigs. We show that with our method, we obtain a higher quality assembly than either only short read or long read assembly. We tried our method with both 454 corrected with Illumina and Ion-Torrent corrected with Illumina and then compared the results. We also combine all three types of contigs (Illumina short-read contigs, 454 long-read contigs and Ion-Torrent long-read contigs) and present our results. We also ran other hybrid assemblers, Celera-CABOG [17,18] and Masurca [19] on the same data and compared the results with ours. The results show that our method improves the resulting assembly and gives better results than Celera-CABOG and Masurca.

1.1.3. Discovery of novel sequence insertions

In the last study, we present Pamir, a new algorithm to efficiently and accurately discover and genotype novel sequence insertions using either single or multiple genome sequencing datasets. Pamir is able to detect breakpoint locations of the insertions and calculate their zygosity (i.e. heterozygous vs. homozygous). To do this, it analyzes

multiple sequence signatures, matches one-end-anchored (OEA) sequences to small-scale *de novo* assemblies of unmapped reads (i.e. orphans) and conducts strand-aware local assembly. We test the efficacy of Pamir on both simulated and real data, and demonstrate its potential use in accurate and routine identification of novel sequence insertions in genome projects.

We give the details of each study in the following chapters. In Chapter 2, we explain the steps we follow to analyze the two whole genome shotgun (WGS) sequence data from the genomes of two human individuals that are generated in two different genome sequencing centers and we go through the results of the analysis. In Chapter 3, we describe our method to improve the genome assemblies using multi-platform sequencing data and we present the comparative results of our method and hybrid assemblers. In Chapter 4, we describe our method on discovering and genotyping the novel sequence insertions in the genomes of one or many individuals.

2. ASSESSING THE ROBUSTNESS OF MASSIVELY PARALLEL SEQUENCING PLATFORMS

Robustness and reproducibility are the essentials of each data that is aimed to be used in clinical diagnostics. Whole-genome array based technologies were introduced to measure the expression levels of many genes in a genome [7]. The main issues blocking large scale applicability of array-based technologies for clinics are these two factors, i.e. robustness and reproducibility. High throughput sequencing (HTS) platforms offer alternative solutions to array based technologies with respect to genotyping. Compared to the array based technology data, HTS data are considered to be more robust and comprehensive. The performance of HTS platforms has already been tested in various studies [9–11], but the questions about robustness of HTS platforms still need to be systematically assessed. It is of crucial importance to obtain accurate single nucleotide polymorphism (SNP), Indel, and structural variation (SV) call sets which means the calls made for specific SNPs or SVs should be only dependent on the actual genotypes of sequenced individuals. They should not be dependent on the platform, location, or time of choice of the HTS study.

The ability of three sequencing platforms, Illumina-Miseq, Roche-454 in terms of read lengths and coverage are assessed in [20]. The performance with respect to coverage and SNP-indel calling of two different sequencing platforms, Illumina and Complete Genomics is compared in [9]. The performance with respect to read lengths and error-rates of the platforms Roche-454, Illumina-MiSeq and Ion-Torrent is compared in [10]. Performances of Ion-Torrent-PGM, Illumina-MiSeq, and Pacific Biosciences in terms of coverage distribution, bias towards GC content ratio, and variant calling are investigated in [21]. A method to obtain high confidence genotypes by integrating multiple datasets from different platforms and using different mapping and variant calling tools is presented in [11]. In addition, fourteen different mapping algorithms have been evaluated with a benchmark on Ion-Torrent sequencing data in [22]. The performances of different platforms or data analysis tools are previously studied in the literature

but investigating the robustness of the data from a single high throughput sequencing platform with sequencing the same DNA twice and analyzing the data with the same mapping and variant calling tools has not yet been established.

Here we investigate a highly used HTS technology in genome sequencing, i.e. Illumina HiSeq platform, in terms of robustness. The reason that we chose Illumina HiSeq2000 to investigate is that, at the time of our analysis, Illumina HiSeq2000 was the most accurate high throughput sequencing platform which has the least sequencing error rate ($\sim 0.1\%$) where other available platforms had higher sequencing error rates (Ion Torrent ($\sim 1\%$ error rate), 454 ($\sim 0.5\%$) and Pacific Biosciences ($\sim 14\%$ error rate)). Also, the cost for sequencing a whole genome with Illumina was the cheapest (0.07$ per Mb) compared to other available platforms (454 (10$ per Mb), Ion-Torrent (1$ per Mb), Pacbio (0.13-0.60$ per Mb)). Providing accurate and cheap sequencing data has made Illumina the most popular sequencing platform for large clinical applications. Therefore, we wanted to investigate the robustness of the data generated by Illumina HiSeq2000.

For our purpose, we resequenced the genomes of two individuals from the Turkish Genome Project (TGP) [23] for a second time. The two genomes were previously sequenced as the first time [23], using the Illumina HiSeq 2000 platform in BGI Shenzhen. We resequenced the same DNA as the second time with the same model of the Illumina HiSeq 2000 platform which is set up at the Turkish Advanced Genomics and Bioinformatics Research Group (TÜBİTAK İGBAM). Although we used the same model sequencing machines, and achieved roughly the same level of coverage, and used the same tools with the same parameters during the analysis, there was significant number of differences between the two trials when we assessed independent analysis of the SNP and Indel calls. Especially, we noticed that roughly 280 thousand of the 3 million SNPs genotyped by the GATK [24] tool in one trial (e.g. BGI) or the other (e.g. TÜBİTAK) are unique to only one callset - which means the reproducibility rate of SNP calls is $\sim 92\%$. We expected to see an improvement in the reproducibility and accuracy on the results if we perform a multisample calling with GATK. It is interesting to see that, the multisample (pooled) calling that jointly analyzes two genome

datasets simultaneously does not seem to substantially improve the reproducibility and thus accuracy of the results. In this study, we question the "sources" of the loss of accuracy in terms of both quality scores and coverage levels in each of the samples. Although the differences between the GATK calls for specific loci on the two samples highly decrease when there is increase in coverage levels in each sample, there are still some cases in which we cannot call the cause of the differences as low coverage or quality score differences.

Our main contribution with this study is a detailed investigation of the causes and types of unique variants within the call sets that are expected to be actually the same. In addition, we try to define strategies to handle such discrepancies if there are more than one WGS dataset generated from the genome of the same donor either with the same or different platforms. With further technological advancements and the cost improvements, sequencing a sample many times can be expected to be prevalent, as storing the data may become more expensive than resequencing the same sample. Here the same donor sample is sequenced twice, to evaluate the outcome of this highly possible situation in the future. For such cases, when multiple WGS sequence of the same donor exists, we state our remarks on how to exploit all the data fruitfully. We describe the standard methods for analysis of all samples that we used in the study in Section 2.1. We present the results of the study and show the shared and exclusive sets of different SNP groups and which regions the exclusive SNPs belong in the genome, in Section 2.2. Finally, in Section 2.3, we present our remarks on the obtained results and conclude.

2.1. Methods

2.1.1. DNA Samples and Ethics Statement

Genomic DNA from two individuals were collected and purified in 2011, only once from the blood of two volunteers for a previously published study [23]. The source, i.e. blood, DNA extraction time and location are the same. Institutional review board permission was obtained from INAREK (Committee on Ethical Conduct

in Studies Involving Human Subjects at the Boğaziçi University) before data collection. All participants including those that are included in this study have already provided consent during the previous study [23]. So, we did not need to get a permission for the second time for this study.

2.1.2. Sequencing

During the Turkish Genome Project [23], the DNAs of the two individuals were already sequenced with Illumina HiSeq 2000 and the first dataset was completed in 2011 at BGI Shenzhen in China. The same DNA samples were sequenced for a second time with another Illumina HiSeq 2000. The second dataset was generated for this study in 2012 at TÜBİTAK İGBAM in Kocaeli Turkey. For the first sequencing, according to the report of BGI, DNA samples were fragmented to 500bp. Paired-end sequencing dataset was produced with the read length of 90bp. For the second sequencing experiment performed at TÜBİTAK, the same protocols are followed, the DNA samples are fragmented to 500bp. The quantified and quality checked gDNA libraries were prepared and paired-end dataset is generated with the read length of 104bp. To eliminate the possible variations in variant calls stemming from the library preps and other wet lab work, the standard variant quality filtering software is used. Hereafter, we refer to the data generated in BGI as S_{1B} (the genomic sequence of the first individual), S_{2B} (the genomic sequence of the second individual) and the data generated in TÜBİTAK for the same first and second individuals as S_{1T} and S_{2T}, respectively.

2.1.3. Alignment, coverage, GC content

We mapped the reads to the human reference genome (NCBI GRCh37) to discover SNP and short Indels. For mapping, we used BWA aligner (version 0-6-2) [25], a widely used aligner in HTS data analysis, with paired-end mode ("sampe") and default parameter values. We followed the general procedure and converted the mapping output to sorted, duplicate-removed, and indexed BAM files using SAMtools [26]. The expected coverage is calculated by dividing the total number of mapped bases to the

number of non-N bases in GRCh37 which can be shown as in Equation 2.1

$$\text{expected coverage} = \frac{\text{num mapped reads} \times \text{read length}}{2,897,310,462} \tag{2.1}$$

Next, we used SAMtools and BEDtools [27] to calculate the effective coverage as in Equation 2.2:

$$\text{effective coverage} = \frac{(\sum_{i=1}^{\text{num\_bases}} \text{Coverage}_i)}{\text{num\_bases}} \tag{2.2}$$

Lastly, FASTQC tool (version 0.10.1) [28] is used to collect the GC content and basic statistics of the genomic sequence data.

2.1.4. Variant calling

2.1.4.1. SNP and Indel detection. After the initial alignment and the PCR-duplicate removal, we re-aligned the Indel-containing reads back to the reference genome using GATK Realigner tool. As the next step, we used the GATK Unified Genotyper tool to obtain the SNP and Indel call sets. As an alternative approach for variant calling, we also used GATK Haplotype Caller for generating SNP and Indel call sets. We investigated if there is a difference between the number of exclusive variants generated by Unified Genotyper and Haplotype Caller. GATK Haplotype Caller is also run for variant calling to compare number of variants with Unified Genotyper output. In the next step, we eliminated likely false positives using the GATK Variant Quality Score Recalibration (VQSR) tool with GATK resource bundle v2.5. VQSR creates a Gaussian mixture model according to the annotation values of a group of high quality variants and according to the model it calculates a new well-calibrated quality score for each variant which increases the sensitivity and specificity. It assigns a new "PASS" or "FAIL" filtering label to the variant according to a threshold value.

As the last step, we used GATK VariantFiltration tool to further remove the low confidence calls. It is the SnpCluster filter that removes SNPs if there are more than 3 SNPs in a 10 bp window. We applied the same variant calling pipeline defined here to each of the four datasets separately: S_{1B}, S_{1T}, S_{2B} and S_{2T}.

2.1.4.2. Long Deletion detection. We discovered long deletions by mapping discordant reads with mrFAST [29] and calling deletions with VariationHunter [30].

2.1.4.3. Pooled SNP and Indel calling. As a second experiment, we tested whether multisample calling, i.e. pooling data from multiple sequencing runs for the same samples, improves callset reproducibility. Mainly we explored if the slight differences in the depth and coverage of the datasets could be improved by merging data for discovery, and if this would improve genotyping accuracy. For this purpose, we applied the same SNP/Indel detection process to both samples by pooling two sequencing datasets generated at BGI and TÜBİTAK. The pooled datasets are named S_{1BT} and S_{2BT} for the first and second samples, respectively.

However, we named the two datasets from the same sample from the pooled calling as if they were generated from different genomes. In the remainder of the paper, we denote the SNP/Indels genotyped within the BGI data from sample S_1 (sample 1) as B_1, and the SNP/Indels genotyped within the TÜBİTAK data from S_1 as T_1 for this experiment. In the same way, we have B_2 and T_2 for the sample S_2 (sample 2) .

2.1.5. Variant annotation

Variant annotation is the process to predict the function of a variant based on its location on the genome. We used ANNOVAR [31](version 2013-02-21) variant annotation tool for annotating SNPs and Indels. It provides the information of whether a variant is located in a functional genomic region like exonic regions or splice sites.

2.1.6. Data Availability

Sequence reads obtained from BGI for the Turkish Genome Project study [23] are already stored to the SRA read archive (SRP021510). Primary run IDs relevant to this study are: SRR839600 for S_{1B} and SRR849493 for S_{2B}. Datasets generated at TÜBİTAK are also stored as "secondary sequencing" with sample IDs SRR2128004 and SRR2128088 respectively under the same SRA archive. The publication [32] is available online. We also released the scripts that we used to map the reads and call the variants at [33]. The resulting VCF files for the variant callsets are stored at [34].

2.2. Results

2.2.1. Read length, coverages, and GC content

The basic statistical analysis results of the datasets such as number of reads, read lengths, expected and effective coverages and GC content rates are given in Table 2.1. The total number of reads generated are of more than 5.5 billion reads which is equivalent to $\sim$ 530 giga basepairs (Gbp). The effective sequence coverage per sample ranges from $29.5X$ to $49.2X$. The reads sequenced at TÜBİTAK (S_{1T} and S_{2T}) (104bp) are 14bp longer than the reads sequenced at BGI (S_{1B} and S_{2B}) (90bp). The GC contents are similar to each other, they are ranged from 39% to 43% (Table 2.1).

Table 2.1. Summary of the sequence datasets.

Dataset	Number of reads	Read length	Expected Coverage	Number of mapped reads	Effective Coverage	GC%
S_{1T}	1,401,819,290	104	45.6X	1,366,858,600	42.3X	42%
S_{1B}	1,394,524,622	90	41.5X	1,272,512,132	37.6X	39%
S_{2T}	934,050,130	104	31.3X	914,763,337	29.56X	43%
S_{2B}	1,793,560,406	90	53.4X	1,688,991,592	49.2X	41%

Sequencing statistics of the two samples (S_1, S_2) sequenced at two different centers. S_{1T} refers to sample S_1 sequenced at TÜBİTAK, where the dataset S_{1B} is generated from the same sample at BGI. Similarly, datasets from sample S_2 are denoted as S_{2T} and S_{2B}.

2.2.2. Callsets and comparisons

2.2.2.1. SNP and Indel discovery. SNP callsets for each individual sample and also for multisample (pooled) sequences are generated (explained in Section 2.1). 4 SNP callsets are named as S_{1T}, S_{1B}, S_{2T}, S_{2B}, and 2 pooled callsets for S_1 and S_2, which are denoted as S_{1BT}, S_{2BT}, were generated. 3 callsets per sample (i.e., S_{1T}, S_{1B}, and S_{1BT} for S_1 and S_{2T}, S_{2B}, and S_{2BT} for S_2) are compared with each other to quantify and characterize any differences. The SNP and Indel statistics of all these 6 callsets for 2 samples which are called by GATK Unified Genotyper are summarized in Table 2.2. The calls for the same callsets which are obtained by Haplotype Caller are summarized in Table 2.4.

2.2.2.2. Long deletions. Total long deletions and novel long deletions compared to 1000 genomes project are shown in Table 2.3.

Table 2.2. (Unified Genotyper) SNP and Indel numbers obtained from the data.

	SNPs		**Indels**	
	Total	**Novel**[1]	**Total**	**Novel**[1]
S_{1T}	3,320,545	40,936	34,407	430
S_{1B}	3,356,829	60,596	132,144	2,076
S_{1BT}	3,340,498	55,408	80,950	1,227
S_{2T}	3,277,433	46,448	56,189	756
S_{2B}	3,346,221	55,753	54,229	529
S_{2BT}	3,393,037	98,383	32,743	502

[1] Compared to dbSNP138

2.2.2.3. Separately generated callsets. In summary, after potential false positive removal with GATK Variant Quality Score Recalibration and VariantFiltration (explained in Section 2.1), we observed approximately 95% agreement between the pairs of SNP callsets generated from both genomes (Table 2.5).The Indel callsets showed a larger discrepancy, where only 18%-68% of each callset were shared with the other two

Table 2.3. Long deletions obtained from the data with mrFAST and VariationHunter.

	Deletions (>50bp)	
	Total	**Novel**[1]
S_{1T}	1,214	15
S_{1B}	1,209	38
S_{1BT}	1,150	40
S_{2T}	1,102	27
S_{2B}	1,008	26
S_{2BT}	932	24

[1] Compared to 1000 Genomes Phase 1

Table 2.4. (Haplotype Caller) SNP and Indel numbers obtained from the data.

	SNPs		**Indels**	
	Total	**Novel**[1]	**Total**	**Novel**[1]
S_{1T}	3,540,735	57,905	614,241	35,624
S_{1B}	3,504,854	58,578	668,779	41,558
S_{1BT}	3,569,295	59,510	739,347	50,617
S_{2T}	3,463,094	60,344	589,891	34,249
S_{2B}	3,539,933	79,869	718,734	44,571
S_{2BT}	3,613,663	72,099	217,365	57,056

[1] Compared to dbSNP138

callsets (Table 2.5).The number of shared and discrepant SNP and Indels generated by Haplotype Caller are shown in Table 2.6.

We studied the underlying sequence content of the discrepancies of novel SNP and Indel calls in detail to understand the causes of different calls from the same genomes. We first downloaded human reference genome annotations for segmental duplications and common repeats from the UCSC genome browser [35] and copy number variation (CNV) callsets from the 1000 Genomes Project [36]. We then counted the number of novel SNP and Indel calls. SNPs and Indels found in segmental duplications, common

repeats, possible CNVs and also the calls discovered in low coverage regions are shown in Figure 2.1A and 2.1C, Figure 2.1B and 2.1D, respectively. We found that 46%-59% of discrepant novel SNP calls intersected with common repeats, and a group of 5%-28% intersected with segmental duplications. In addition, a 3%-5% of the discrepant calls were found within possible CNV regions reported in the 1000 Genomes Project [36], and 0.3%-0.8% were discovered in low coverage regions ($<$ 5X). We also analyzed the discrepant Indel calls which showed similar distribution. The distribution of discrepant Indel calls to the regions are shown in Figure 2.1B and 2.1D. The majority of discrepant calls were found to be within Alu and L1 repeats (Tables 2.7 and 2.8). The discrepant calls within satellites and low complexity repeats were negligible. In addition, a close look to Alu and L1 subfamilies revealed that the number of discrepant calls peaked at $\sim$10% sequence divergence from consensus sequences, also showing negligible differences at recent and distant mobile element insertion loci (data not shown). Both of these observations can be explained by low mapping quality within these regions, causing the VQSR algorithm to filter out such calls.

We studied the significance of the discrepant SNP and Indels closely in terms of predicted functionality. The distribution of discrepant SNP and Indels in terms of their functionality is shown in Table 2.9. We found that 88%-95% of the discrepant SNP calls mapped to intergenic and intronic regions where a 3.5%-4.5% were predicted to be within coding exons, and noncoding RNAs (ncRNAs). Intergenic regions in the DNA exist in between genes. The reason they are called "junk DNA" is that currently they do not have any known function and they are part of noncoding DNA. Intronic regions exist in between coding exons within the genes. They are the intervening sequences which are also part of noncoding DNA and are removed by RNA splicing during the creation of mature RNA. Intergenic and intronic regions has no direct role on protein coding. Coding exons are the DNA parts within genes which directly code DNA. Noncoding RNAs are transcribed from DNA but are not translated into proteins. Some ncRNAs have important roles on regulating gene expression. The noncoding intergenic and intronic regions are less interesting compared to coding exons. The importance of long ncRNA on regulating genes is discovered more in recent years [37]. Indels showed similar distribution properties, where only 0-3 of them were predicted to incur

frameshifts. The frameshift insertions or deletions have important affect on protein coding, so they are possible disease causing mutations.

Table 2.5. (Unified Genotyper) Comparisons of total and novel SNP and Indel callsets generated from the genomes of S_1 and S_2. S_{1B}, S_{1T}, S_{1BT}: S_1 calls from BGI, TÜBİTAK, and pooled datasets; S_{2B}, S_{2T}, S_{2BT}: S_2 calls from BGI, TÜBİTAK, and pooled datasets, respectively.

	SNPs		**Indels**	
	Total	**Novel**	**Total**	**Novel**
$\lvert S_{1B} \cap^{*} S_{1T} \cap S_{1BT} \rvert$	3,167,254 (90.0%)	36,273 (49.3%)	23,293 (15.2%)	232 (9.1%)
$\lvert (S_{1B} \setminus^{\#} S_{1T}) \setminus S_{1BT} \rvert$	75,839 (2.2%)	16,073 (21.9%)	67,478 (43.9%)	1,239 (48.6%)
$\lvert (S_{1T} \setminus S_{1B}) \setminus S_{1BT} \rvert$	56,906 (1.6%)	1,444 (1.9%)	3,525 (2.3%)	56 (2.2%)
$\lvert (S_{1BT} \setminus S_{1B}) \setminus S_{1T} \rvert$	22,737 (0.6%)	8,896 (12.1%)	11,647 (7.6%)	300 (11.7%)
$\lvert (S_{1B} \cap S_{1T}) \setminus S_{1BT} \rvert$	29,807 (0.9%)	615 (0.8%)	1,476 (0.9%)	26 (1.0%)
$\lvert (S_{1B} \cap S_{1BT}) \setminus S_{1T} \rvert$	83,929 (2.4%)	7,635 (10.4%)	39,897 (26.0%)	579 (22.7%)
$\lvert (S_{1T} \cap S_{1BT}) \setminus S_{1B} \rvert$	66,578 (1.9%)	2,604 (3.5%)	6,113 (3.9%)	116 (4.5%)
$\lvert S_{2B} \cap S_{2T} \cap S_{2BT} \rvert$	3,164,900 (90.0%)	42,518 (40.0%)	12,823 (13.4%)	93 (6.7%)
$\lvert (S_{2B} \setminus S_{2T}) \setminus S_{2BT} \rvert$	40,492 (1.2%)	4,899 (4.6%)	22,599 (23.6%)	258 (18.7%)
$\lvert (S_{2T} \setminus S_{2B}) \setminus S_{2BT} \rvert$	62,748 (1.8%)	46,415 (43.8%)	34,980 (36.5%)	581 (42.2%)
$\lvert (S_{2BT} \setminus S_{2B}) \setminus S_{2T} \rvert$	62,029 (1.8%)	2,314 (2.2%)	3,567 (3.7%)	219 (15.9%)
$\lvert (S_{2B} \cap S_{2T}) \setminus S_{2BT} \rvert$	12,972 (0.4%)	251 (0.2%)	5,420 (5.6%)	35 (2.5%)
$\lvert (S_{2B} \cap S_{2BT}) \setminus S_{2T} \rvert$	127,857 (3.6%)	8,085 (7.6%)	13,387 (13.9%)	143 (10.4%)
$\lvert (S_{2T} \cap S_{2BT}) \setminus S_{2B} \rvert$	37,532 (1.1%)	1,365 (1.3%)	2,966 (3.1%)	47 (3.4%)

* Cardinality of the intersection: $|A \cap B|$ = The number of SNPs or Indels shared by set A and set B.

$^{\#}$ Cardinality of the difference: $|A \setminus B|$ = The number of SNPs or Indels that set A has and set B does not have.

Table 2.6. (Haplotype Caller) Comparisons of total and novel SNP and Indel callsets generated from the genomes of S_1 and S_2. S_{1B}, S_{1T}, S_{1BT}: S_1 calls from BGI, TÜBİTAK, and pooled datasets; S_{2B}, S_{2T}, S_{2BT}: S_2 calls from BGI, TÜBİTAK, and pooled datasets, respectively.

	SNPs		**Indels**	
	Total	**Novel**	**Total**	**Novel**
$\lvert S_{1B} \cap S_{1T} \cap S_{1BT} \rvert$	3,373,868 (91.7%)	43,693 (58.8%)	552,114 (72.8%)	22,090 (36.3%)
$\lvert (S_{1B} \setminus S_{1T}) \setminus S_{1BT} \rvert$	36,182 (0.9%)	7,005 (9.4%)	7,863 (1.0%)	6,189 (10.2%)
$\lvert (S_{1T} \setminus S_{1B}) \setminus S_{1BT} \rvert$	55,145 (1.5%)	6,663 (8.9%)	9,729 (1.3%)	3,735 (6.1%)
$\lvert (S_{1BT} \setminus S_{1B}) \setminus S_{1T} \rvert$	25,347 (0.7%)	2,418 (3.3%)	27,621 (3.6%)	5,919 (9.7%)
$\lvert (S_{1B} \cap S_{1T}) \setminus S_{1BT} \rvert$	18,223 (0.4%)	1,015 (1.4%)	794 (0.1%)	235 (0.4%)
$\lvert (S_{1B} \cap S_{1BT}) \setminus S_{1T} \rvert$	76,581 (2.1%)	6,865 (9.2%)	108,008 (14.3%)	13,044 (21.5%)
$\lvert (S_{1T} \cap S_{1BT}) \setminus S_{1B} \rvert$	93,499 (2.5%)	6,534 (8.8%)	51,604 (6.8%)	9,564 (15.7%)
$\lvert S_{2B} \cap S_{2T} \cap S_{2BT} \rvert$	3,334,025 (89.9%)	46,783 (45.1%)	543,893 (66.4%)	22,332 (34.0%)
$\lvert (S_{2B} \setminus S_{2T}) \setminus S_{2BT} \rvert$	35,153 (0.9%)	18,073 (17.4%)	4,807 (0.6%)	1,762 (2.7%)
$\lvert (S_{2T} \setminus S_{2B}) \setminus S_{2BT} \rvert$	52,188 (1.4%)	8,034 (7.7%)	16,981 (2.1%)	6,611 (10.1%)
$\lvert (S_{2BT} \setminus S_{2B}) \setminus S_{2T} \rvert$	43,596 (1.2%)	10,903 (10.5%)	54,639 (6.7%)	9,291 (14.2%)
$\lvert (S_{2B} \cap S_{2T}) \setminus S_{2BT} \rvert$	5,797 (0.1%)	600 (0.5%)	687 (0.1%)	175 (0.3%)
$\lvert (S_{2B} \cap S_{2BT}) \setminus S_{2T} \rvert$	164,958 (4.4%)	14,413 (13.9%)	169,347 (20.7%)	20,302 (30.9%)
$\lvert (S_{2T} \cap S_{2BT}) \setminus S_{2B} \rvert$	71,084 (1.9%)	4,927 (4.7%)	28,330 (3.5%)	5,131 (7.8%)

* Cardinality of the intersection: $|A \cap B|$ = The number of SNPs or Indels shared by set A and set B.

Cardinality of the difference: $|A \setminus B|$ = The number of SNPs or Indels that set A has and set B does not have.

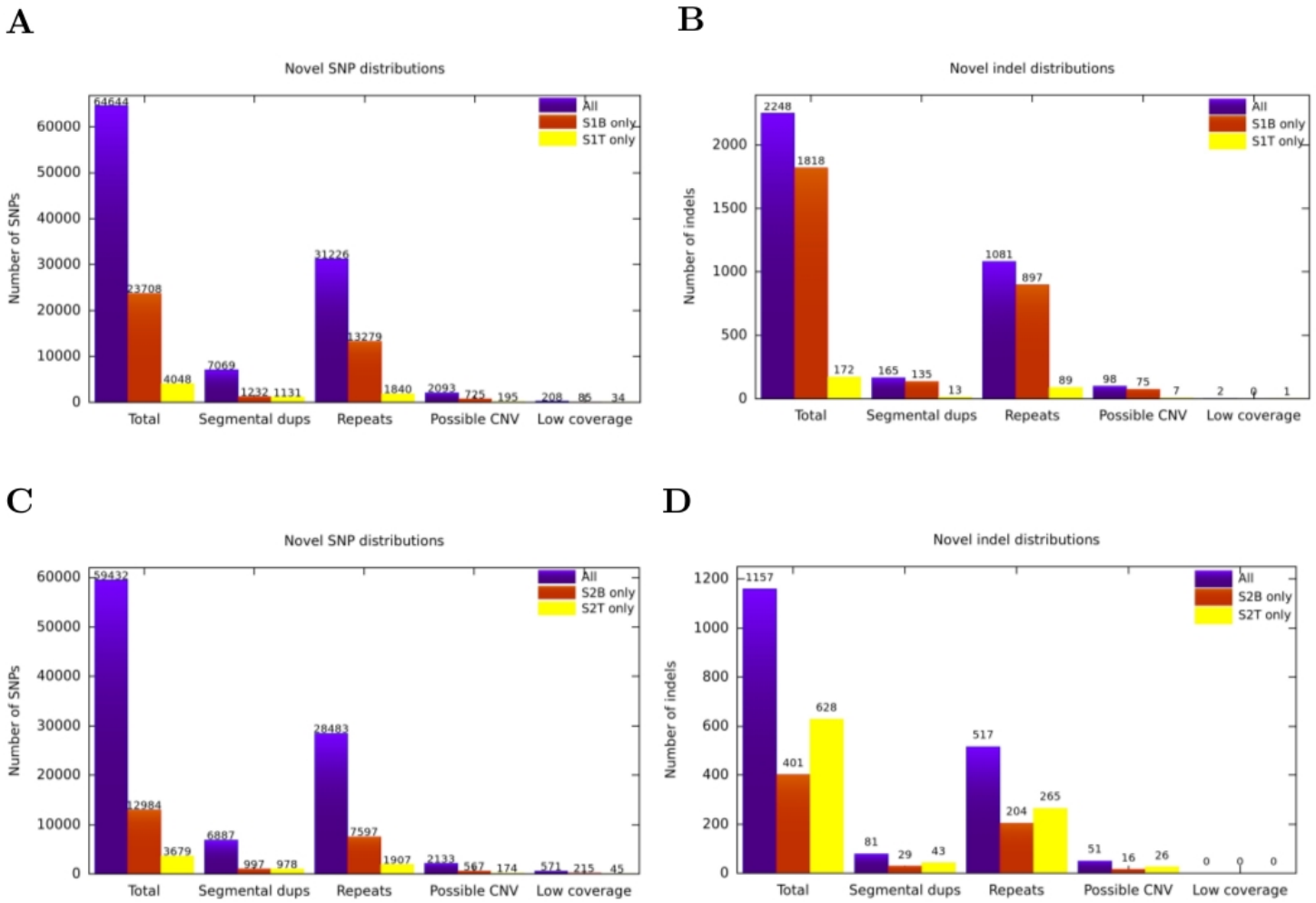

Figure 2.1. Underlying sequence content of novel SNP and Indel calls. A) SNPs and B) Indels in the genome of S_1. C) SNPs and D) Indels in the genome of S_2.

Table 2.7. Detailed view of novel SNP and Indel distributions of S_1 that map to common repeats.

	Novel SNPs at common repeats			**Novel Indels at common repeats**		
	All S_1	**S_{1B} only**	**S_{1T} only**	**All S_1**	**S_{1B} only**	**S_{1T} only**
Total	31,226	13,279	1,840	1,081	897	89
SINE/Alu*	8,911	4,175	706	204	196	5
LINE/L1	8,779	3,581	332	415	330	33
LTR/ERV	5,370	2,022	263	84	74	4
Low compl.	429	196	55	63	41	11
Satellite	237	89	14	9	7	0
Simple rep.	1,605	1,011	312	151	118	27
Other	5,895	2,205	158	155	131	9

* SINE/Alu: Short Interspersed Nuclear Elements

LINE: Long Interspersed Nuclear Elements

LTR/ERV: Long Terminal Repeats/Endogenous Retroviruses

2.2.2.4. Pooled BGI vs Pooled TÜBİTAK. The number of shared and discrepant SNP and Indel calls of multisample (pooled) calling are shown in Table 2.10. This strategy showed a better correspondence between the two datasets, reducing the contradicting call rate to 0.1%-0.8%. The number of shared and discrepant SNP and Indel calls of pooled calling with Haplotype Caller are also shown in Table 2.11.

2.3. Discussion

With the improvements in cost efficiency, speed, and analysis algorithms, HTS platforms are now being considered to be used routinely as part of clinical diagnosis. This assumption started a pilot project called ClinSeq [38] that aims to investigate the strength and potential pitfalls of using HTS data in the clinic. However, the HTS technologies continue to evolve and new platforms are introduced almost every year. This fact, coupled with changes and updates of algorithms to analyze HTS data, raises questions about the maturity and robustness of HTS platforms for accurate discovery

Table 2.8. Detailed view of novel SNP and Indel distributions of S_2 that map to common repeats.

	Novel SNPs at common repeats			**Novel Indels at common repeats**		
	All S_2	**S_{2B} only**	**S_{2T} only**	**All S_2**	**S_{2B} only**	**S_{2T} only**
Total	28,483	7,597	1,907	517	204	265
SINE/Alu	9,499	4,048	507	71	45	24
LINE/L1	7,396	1,331	511	208	71	112
LTR/ERV	4,360	434	221	66	20	38
Low compl.	653	399	59	32	17	12
Satellite	260	61	29	0	0	0
Simple rep.	1,489	784	410	54	26	27
Other	4,826	540	170	86	25	52

* SINE/Alu: Short Interspersed Nuclear Elements

LINE: Long Interspersed Nuclear Elements

LTR/ERV: Long Terminal Repeats/Endogenous Retroviruses

and genotyping of genomic variants. The performance of different sequencing platforms in terms of read lengths, coverage and GC ratio have been assessed in several studies but the robustness of them have not yet been analyzed.

In an effort to answer this question, we analyzed the genomes of two individuals, each sequenced twice using the same widely used technology, Illumina HiSeq2000, albeit at different locations. We chose Illumina HiSeq2000 to investigate because it is the most popular sequencing platform for large clinical applications by generating accurate reads with low costs. Since our aim was to investigate the maturity of sequencing platforms in this study, we used the same tools to characterize both single nucleotide and short Indel variants. Under the assumption of 100% robustness, one would expect to characterize the same set of variants in both sequencing datasets from the same genomes. However, this is not what we found.

We believe multiple factors contribute to this effect. First, since the library preparation is different, one may expect difference in GC% bias, as clearly seen in Table 2.1.

Table 2.9. Distribution of discrepant novel SNP-Indels of S_1 and S_2 over gene regions.

	Novel discrepant calls of S_1				**Novel discrepant calls of S_2**			
	S_{1T}		**S_{1B}**		**S_{2T}**		**S_{2B}**	
	SNP	**Indel**	**SNP**	**Indel**	**SNP**	**Indel**	**SNP**	**Indel**
Total	4,048	172	23,708	1,818	3,679	628	12,984	401
intergenic	2,191	107	13,451	1,029	2,261	358	6,470	249
intronic	1,506	50	8,899	694	1,196	233	5,016	126
upstream	62	2	139	10	34	2	467	4
downstream	44	1	144	8	28	2	89	3
UTR5	33	0	36	1	5	1	228	1
UTR3	29	3	199	17	21	5	96	5
exonic nonsyn	26	0	129	0	5	0	131	0
exonic syn	24	0	47	0	7	0	42	0
exonic stopgain	0	0	5	0	0	0	0	0
exonic unknown	0	0	1	0	0	0	4	0
exonic	0	0	1	0	0	0	0	0
ex. frmshift del[1]	0	0	0	1	0	1	0	0
ex. nonfrmshift del	0	0	0	1	0	0	0	0
ex. nonfrmshift ins	0	0	0	1	0	0	0	0
splicing	1	0	13	1	2	0	31	0
ncRNA intronic	114	9	609	55	116	26	357	12
ncRNA exonic	17	0	33	0	4	0	39	1
ncRNA UTR5	1	0	1	0	0	0	8	0
ncRNA UTR3	0	0	0	0	0	0	6	0
ncRNA splicing	0	0	1	0	0	0	0	0

[1] ex. frmshift del: exonic frameshift deletion

This leads to differences in read depth over different regions of the genome, which in turn causes discrepancies in variation calls. The GC% effect can also explain the over-representation of repeats and segmental duplications in terms of SNP discrepancies, as common repeats are high in GC content (41.45% GC within common repeats vs 40.33% GC in unique regions), together with difficulties in mapping to repeats and duplica-

Table 2.10. (Unified Genotyper) Comparisons of total and novel SNP and Indel intersections of B_1 vs. T_1 and B_2 vs. T_2. B_1, T_1:pooled S_1 calls from BGI and TÜBİTAK datasets; B_2, T_2:pooled S_2 calls from BGI and TÜBİTAK datasets, respectively.

	SNPs		**Indels**	
	Total	**Novel**	**Total**	**Novel**
$\|B_1 \cap T_1\|$	3,308,870	41,289	79,948	1,195
$\|B_1 \setminus T_1\|$	25,857	13,536	651	17
$\|T_1 \setminus B_1\|$	5,771	483	351	15
$\|B_2 \cap T_2\|$	3,321,318	51,526	32,391	468
$\|B_2 \setminus T_2\|$	70,068	46,592	121	11
$\|T_2 \setminus B_2\|$	1,651	265	231	23

Table 2.11. (Haplotype Caller) Comparisons of total and novel SNP and Indel intersections of B_1 vs. T_1 and B_2 vs. T_2. B_1, T_1:pooled S_1 calls from BGI and TÜBİTAK datasets; B_2, T_2:pooled S_2 calls from BGI and TÜBİTAK datasets, respectively.

	SNPs		**Indels**	
	Total	**Novel**	**Total**	**Novel**
$\|B_1 \cap T_1\|$	3,551,861	57,010	735,208	49,637
$\|B_1 \setminus T_1\|$	5,653	1,164	1,396	346
$\|T_1 \setminus B_1\|$	11,781	1,336	2,743	634
$\|B_2 \cap T_2\|$	3,595,114	69,416	789,834	55,740
$\|B_2 \setminus T_2\|$	11,140	1,722	3,687	719
$\|T_2 \setminus B_2\|$	7,409	961	2,688	597

tions. Second, although the make and model of the sequencing instruments are the same, they are individually different machines, which may account for slight differences in base calling errors. Third, mapping biases against repeats and duplications incur additional problems in terms of mapping and calling. In our publication [32] on this study we mentioned that "we used the same mapping and calling tools with the same parameters for all datasets in this study, therefore the tools should not be the reason for discrepancies." It was before another very similar study [39] in which the authors found out that the mapping tool BWA has a bias against the read orders in FASTQ file.

The reads from repeat regions map to different locations if they are reshuffled. They also found out that even if the same alignment file is used for variant calling, GATK Haplotype Caller generates different callsets. After this, we cannot state our previous note about the expectation of "the tools should not be the reason for discrepancies"; actually they can be the reason.

Sequencing machines, alignment and genomic variant discovery and genotyping algorithms change rapidly, and one must be careful when interpreting the results. Orthogonal studies with more than one platform [16] or more than one sequencing with the same platform can be considered to have better sensitivity and specificity for variant calls. Although orthogonal methods are needed for definitive validations, we suggest that when there are more than one dataset, one should use all the available data for higher accuracy. Here we demonstrated potential problems that may arise within HTS-based studies. Discrepancies between call sets generated from the same genomes may be complementary false positives and false negatives in each callset, in addition to common genotyping errors. Luckily, much of the differences were found within non-genic regions, intronic regions and common repeats, which are of less importance for most studies.

3. IMPROVING GENOME ASSEMBLIES WITH MULTI-PLATFORM SEQUENCE DATA

The great ability of high throughput sequencing (HTS) platforms to generate huge amounts of data made large-scale sequencing projects possible. This reduced the attention towards traditional Sanger sequencing. The attention moved towards large scale sequencing projects for which Sanger sequencing is pretty expensive. On the other hand HTS is getting cheaper from day to day. HTS is cost effective for data production but enormous amount of data needs to be stored and to be analyzed which imposes the increased cost for storing, data processing, and computational burden. In addition, the data obtained from HTS technologies is in fact of lower quality compared to Sanger sequencing. The error rates are greater and the read lengths are shorter for most platforms. One of the main algorithmic problems in analyzing HTS data is the *de novo* assembly: i.e. "stitching" billions of short DNA strings into a collection of larger sequences, ideally the size of chromosomes. However, accurate *de novo* assembly with no gaps and no errors is still missing due to many factors. Some of these factors are sequencing errors in basepair level, the short read and fragment (paired-end) lengths, and the complex and repetitive nature of most genomes. There are several assembly algorithms developed for assembling the data generated with different high throughput sequencing technologies, and also there are some that can use data from more than one sequencing technology but still the assemblies are far from being perfect. There is still a need for computational approaches to improve draft assemblies.

There are briefly three classes of assemblers mainly used to do genome assembly [14] : (i) greedy assemblers [40–42], (ii) overlap-layout-consensus (OLC) graph based assemblers [17, 43–45], and (iii) de-Bruijn graph based assemblers [46–51]. In addition, although they use one of the de-Bruijn or OLC graph based assembly, there are also assemblers called hybrid assemblers which combine data from different platforms.

3.1. Greedy Assembly

Greedy assemblers follow a greedy approach in the sense that given one read or contig, at each step the assembler adds one more read or contig with the largest overlap. The biggest problem of greedy assemblers is that they can get stuck at local maxima. They use more computational sources and are usually slow compared to other assemblers too. Therefore they are not feasible for large genome assemblies and are generally used for small genome assemblies. The greedy assembler SSAKE [40] at first was working on only single end reads, later on it has been improved to exploit paired-end reads too. It uses a lookup hash table indexed by unique sequence reads with their occurences in the file. The reads and their reverse complements are indexed by their first 11 bases at the $5'$ end prefixes and they are sorted according to their decreasing number of occurences. SSAKE iteratively searches the reads that overlap prefix-to-suffix at least above a threshold, to form an assembly. It stops when all possibilities of $3'$ extension are finished and also the reverse complementary extensions for the $5'$ end. SCHARCGS [41] also works on unpaired short reads. It is similar to SSAKE, but it also applies pre-processing and post-processing to the algorithm. In the pre-processing step, SCHARCGS filters the erroneous reads according to coverage. After pre-processing according to different settings, 3 sets of reads are obtained and each set is assembled by iterative contig extension. In the post-processing step, the three contig sets are merged by sequence alignment. Another greedy assembler VCAKE [42] builds two hash tables from a group of reads called bin and set. The extension of seeds (first eleven bases) in the set are used to search bin. Its superiority to the previous assemblers is its ability to allow one mismatch after the first eleven bases during contig extension.

3.2. Overlap Layout Consensus (OLC) Graph Based Assembly

OLC graph based assemblers were developed to work on long Sanger data and then it is optimized for large genomes. So, OLC graph based assemblers work well usually on the long reads. They first generate all-against-all pairwise alignments (both forward and reverse complement orientations) which construct the “overlap” part. They

use seed and extend heuristic algorithm while making pairwise read comparison. In seeded algorithms, short, unique, fixed length sequences of the DNA (k-mer content) are pre-computed across all reads, reads that share k-mers are identified, alignments are computed using the k-mers as alignment seeds. Then, they build the graph in which the nodes represent the reads and the edges represent the overlaps between the reads. The graph may also contain additional information like forward and reverse complement of reads, the variables to distinguish the $3'$ and $5'$ end of the read, the length of the read and the type of the overlap. After finding a "layout" on the graph, they obtain the "consensus" assembly by following a Hamiltonian path on the graph in which one visits each node exactly once. Finding a Hamiltonian Path is an NP-complete problem [52]. The time required to solve Hamiltonian path increases exponentially with the problem size. The consensus sequence is determined after contig generation and scaffolding. Ideally there is only a single scaffold, but there may be gaps between scaffolds resulting in a partial genome [8, 53].

Newbler is a popular OLC assembler [43] used with 454 data. It runs OLC twice. The first OLC builds unitigs from reads. Unitigs are the preliminary high-confidence conservative contigs. The second OLC uses the unitigs and builds a contig layout based on pair-wise overlaps between them. Unitigs can also be splitted, which may split the reads too, which may be chimera or from repeat region.

The Celera Assembler [17] is also built for Sanger data but then revised for 454 data. The revised version of Celera is called CABOG [18] which finds overlaps between compressed seeds. It reduces the homoplymers to single bases to solve the uncertainty in the data. Initially, it does not use the reads which are substrings of other reads which are possible false overlaps. It compares each read with a set of overlapping reads and detects a sequencing error at each contradiction to the majority, but instead of fixing the error, it assigns an error rate to it. According to a user-defined error-threshold, among the overlapping reads that pass the threshold and the length limit, it selects the longest overlap per read end. From the best overlaps and reads it builds an overlap graph and finds the unitigs from maximal simple paths. Then another graph is built from unitigs and paired-end constraints. From this new graph, unitigs construct contigs

and contigs construct scaffolds. Consensus sequence is obtained by multiple sequence alignments from scaffold layouts and read sequences.

SGA (String Graph Assembler) is another OLC graph assembler which uses compressed data structures [44, 54]. It uses FM-index on sequence reads, it is parallelizable and it requires low memory. SGA first applies pre-processing and filters the reads. After filtering it constructs FM-index from the remaining reads. It then applies error correction using k-mer frequencies. After reindexing the corrected reads, it removes the duplicated sequences and low-quality sequences. From the remaining strings it builds the OLC graph. Then the graph computations follow.

Edena [55] and Shorty [56] are OLC graph assemblers that work on short reads from Solexa and SOLID. Edena is designed for single reads. It gets rid of repeated reads, and finds error-free overlaps. It uses overlap transitive reduction algorithm to get rid of individual overlaps by getting rid of bubbles and spurs [31]. Shorty accepts a group of long reads to have them as seeds and to align short reads onto them. It continues like this and at the end it uses contigs as seeds to construct larger contigs.

Hapsembler is another OLC graph based assembler that uses paired-end short read sequences on an OLC graph [45]. It is not a simple OLC graph, it combines OLC approach with a haplotype-aware Bayesian error correction approach. It also uses a novel matepair graph which is used in reconstructing the genome with paired sequencing reads. Hapsembler first filters the reads in pre-processing and then applies pairwise sequence alignment on the reads. The obtained overlaps are used in Naïve Bayes error correction algorithm to get rid of the sequencing errors. Hapsembler constructs a bi-directed overlap graph with these corrected reads. This graph gives the tiling paths between paired-reads. Then the graph is simplified and the paths between mate pairs are found. A mate-pair graph is built which is constructed with nodes as mate-pairs and edges as possible overlaps between mate-pairs. The maximal path in the mate-pair graph gives the contigs.

3.3. De-Bruijn Graph Based Assembly

De-Bruijn graph based assemblers are designed for assembling short reads. In a de-Bruijn graph, nodes are the fixed length string (k-mers) where k is shorter than the read length and the edges are the suffix-to-prefix overlaps. There is no need for calculating all-against-all pairwise alignments. De-Bruijn graphs also do not store individual reads or overlaps, it stores just the k-mers. In genome assembly de-Bruijn graphs are generally used with edges as k-mers which are unique subsequences within longer reads and nodes as the overlaps between consecutive k-mers. Another type [48] may use it with nodes as k-mers and edges as the overlaps between k-mers. K-mer graphs are very useful when there is large amount of short reads. The Eulerian path in the graph generates the consensus sequence. Eulerian path visits each edge exactly once which is easier than finding an Hamiltonian path. When used for WGS assembly, the k-mer graph represents all of the input reads. Each read is in a path and the set of overlapping reads results in a common path. De-Bruijn graph based assembly is more sensitive to sequencing errors than OLC graph based assembly and they cannot detect repeats longer than the read length but OLC graph assembly can detect longer repeats [8]. Sequencing errors, DNA's double stranded structure, repetitive and palindromic structure are the main problems that de-Bruijn graph based assembly faces which generate tips, bubbles and chimeric structures on the graph. Pre-processing, weighting the graph edges, and eliminating the low-weight edges are some solutions to these problems [8, 57, 58].

Euler is a de-Bruijn graph assembler first developed for Sanger reads then modified for short 454 reads, and then modified for short unpaired and then paired Illumina reads too [46, 59]. In Euler, at first pre-processing is done to detect and note erroneous k-mers. The k-mers which are repeated in several reads are supposed to be the most true k-mers. The unique k-mers are supposed to be the most erroneous k-mers. Low-frequency k-mers are either excluded or corrected. Correction can destroy true k-mers with low coverage and a read can be signed as incorrect if its k-mers never occur together in another read. Pre-processing may also hide the true polymorphism. Then a k-mer graph is built with the rest of the corrected reads. It tries to solve the

problems caused by sequencing errors and genomic repeats by some graph operations. For example, it tries to resolve the repeats of length between k and the read length by mapping the reads through the k-mer graph. Reads including a repeat can solve one copy of the repeat by pulling out one piece of the frayed rope pattern [8]. Euler exploits paired-end reads by mate threading, which is behaving the paired reads as one long read which has missing bases in the middle. This helps resolving repeats. Euler applies graph simplifications according to low or high coverage. It deletes the tips and also highly possible repetitive edges, so breaks the contig at repeat boundaries [8].

Another de-Bruijn graph based assembler Velvet [48] follows a different strategy by storing a group of overlapping k-mers by $k-1$ nucleotides in the nodes of the graph. The information is hold on a sequence which consists of the prefix-suffix of the k-mers. Each node also contains information for the reverse complements of the k-mers. The nodes are connected with directed edges. Velvet builds a hash table consisting of k-mer and the id of the read k-mer exists and the k-mer's location on the read. Another database of Velvet contains the information of the k-mers that are overlapping by subsequent reads, for each read. If the k-mer does not overlap with other reads then a node is created for it. Graph is constructed from these k-mers. The graph is followed by the "roadmaps" existing in the first hash table, the edges are put or incremented according to the correspondence between the k-mers and the nodes. After construction, the graph is simplified by error removal without losing any information in an order: first the tips then the bubbles and then the erroneous connections due to cloning errors or to distant merging tips. The tips are removed according to the length and minority count. The bubbles are removed with the tour bus algorithm. The redundant paths are detected by Dijkstra-like breadth-first search. In the tour bus algorithm, some markers are put along both paths node by node. Two paths are merged by comparing the minority node to the consensus node by local sequence alignment. The information on the node is also mapped onto the majority node. Tour Bus algorithm modifies the path and corrects the graph. After this step the erroneous connections are removed according to coverage. After all, with the help of Tour Bus algorithm, the unique regions with low coverage are not removed, they are left as long nodes [48, 60].

ABySS is another de-Bruijn graph assembler which tries to solve memory problems during the assembly of whole mammalian genomes with de-Bruijn graphs [8, 47]. ABySS is a parallel assembler for short reads, which can distribute the k-mer graphs and graph computations to a grid. ABySS removes the tips and bubbles sequentially like Velvet. ABySS uses a distributed de-Bruijn graph and for that it needs two information, the location of the k-mer and the adjacency information between k-mers independent of the location of the k-mer. The location of the k-mer is computed with a hash function by assigning 0, 1, 2, 3 to the bases A,C,G,T. The adjacency information is stored on the edges. A k-mer has eight possible neighbors. After the k-mers are put on the de-Bruijn graph, the adjacency information of the k-mers with the neighbors are computed and set according to the neighbors. After removing tips and bubbles, node merging procedure is applied by removing ambiguous edges and merging the remaining unambiguous edges. To merge the initial contigs paired-end information is used.

ALLPATHS de-Bruijn graph based assembler exploits multiple length paired-end reads [50, 61, 62]. ALLPATHS first computes the unipaths by behaving the reads as unpaired. Then using these unipaths and paired-end reads, it selects the seeds. Seeds are the unipaths which are selected to build the assembly around. Ideal seeds must be long and must not include more than one copy number. Then, ALLPATHS tries to build neighborhoods around the seeds. The neighborhood is the extended version of the seed (10kb on each side). Then the neighborhood is assembled. For the assembly first, the unipaths which are low in copy number are described in the neighborhood. Then, two sets of reads are constructed: the set near the seed and the set near and outside the seed. After this step, longer reads are obtained. Then all closures of "merged short-fragment pairs" [50] are computed. These closures cover all neighborhood, but also there are some wrong closures. The obtained closures are added together which gives a local assembly. Then, by adding local assemblies to each other it obtains the global assembly. At the end, it edits the assembly by removing ambiguous parts.

SOAPdenovo is another de-Bruijn graph assembler which applies pre-assembly error correction to reduce the memory usage for large datasets [63, 64]. Then, it builds the de-Bruijn graph with the short reads from both ends of the clones and then it

removes the tips, chimeric connections, the bubbles and repeat connections. After obtaining the contigs it re-aligns the reads to the contigs for scaffolding. It uses paired-end information to fill the gaps between the contigs.

Cortex [65] uses colored-de-Bruijn graphs to detect genetic variation in multiple individuals by coloring the nodes and edges in different colors according to the samples. To detect variants of an individual against a reference genome, the reference sequence is colored as red, and the sample sequence is colored as blue on the graph and one can find out the polymorphisms which generate bubbles diverging from the reference and repeat structures which are also observed as bubbles in the reference. One can see different structures by putting multiple individuals all in different colors in the graph. It is capable of limited number of individuals and it has high computational requirements.

3.4. OLC vs. De-Bruijn

In de-Bruijn graphs, there is no need for pairwise sequence alignment to detect the perfect overlaps. De-bruijn graphs also do not store individual reads or overlaps, it stores just the k-mers. In de-Bruijn raphs, one tries to find an Eulerian path which visits every edge exactly once. It is easier than finding an NP-complete Hamiltonian path, which visits every node exactly once in OLC graphs. The disadvantage over OLC graph is de-Bruijn graph is more sensitive to sequencing errors. Overlap graphs can detect repeats longer than a read but de-Bruijn (k-mer) graphs can detect only perfect short repeats. The length of the found repeat in the de-Bruijn graph has to be k or more but less than the read length [8]. Overlaps in the de-Bruijn graphs are limited to the k, which is not the case in OLC graphs. There might be multiple Eulerian paths in de-Bruijn graphs which makes assembly more difficult. Another disadvantage of de-Bruijn graph based assembly is, the de-Bruijn graph assemblers do not have distinct stages as OLC assembler do, so it is difficult to modify the de-Bruijn graph assemblers according to users' needs.

De-Bruijn graph based assemblers are proper for assembling short-reads because short reads are highly accurate and the sequencing error rate of short reads are very low ($< \%0.1$). De-bruijn graph based assemblers are not proper for assembling long reads because long reads have higher sequencing error rate and this will cause a de-Bruijn graph based assembler generate more than one assembly which will lead to ambiguity. OLC graph based assemblers are proper for long read assembly because they use overlap lengths instead of k-mers which eliminates any duplications caused by sequencing errors in the resulting assembly.

3.5. Hybrid Assembly

Several assemblers use more than one dataset for better assembly construction, which are called hybrid assemblers [18,19,66–69]. Hybrid assembly is expected to lower the number of errors in the reads and so contigs [67].

PBcR (PacBio corrected Reads) [66] is developed as part of the Celera assembler [17]. The method corrects the long single molecule PacBio reads which have high error rate with the highly accurate short reads. It maps the short reads onto the long reads to obtain highly accurate consensus sequence. The corrected reads are then assembled by themselves or used with other data. LSC [70] is another tool to apply error correction on PacBio data. It first compresses both short and long reads by homopolymer compression (HC). It then applies a quality control on short reads. Then it aligns short reads onto long Pacbio read contigs to reduce the error rate on the long reads. At the end decompression transformation is done to get the last corrected long Pacbio reads. PBcR and LSC both require high computational resources and time even on small genomes. Cerulean is another hybrid assembler which exploits both short and long reads [69]. It first assembles the short reads and obtains a consensus assembly. Then, to resolve the repeats on this assembly, it maps the long reads onto the assembly. Cerulean takes an ABySS assembly graph of paired-short reads and the mapping of long reads onto these assembled contigs. It gives a simplified assembly graph of which the nonbranching path gives the scaffold of the genome. Ray is another hybrid assembler which uses both 454 and Illumina reads on an annotated de-Bruijn graph [67]. Each

k-mer has a c-confident score and represented with a node of the graph. C-confident score means the coverage of that k-mer, how many times it occurs in the reads. The edges of the graph represent 1-confident $(k+1)-$mers. Which means $(k+1)-$mer occurs at least once in the reads. Ray does not search for an Eulerian path, instead it uses offset functions, which gives a value to each path. It defines seeds and generates contigs from each seed as long as it is possible. Using heuristics with the offset values it finds the best contig paths on the graph. Another hybrid assembler Masurca uses Illumina, 454 and Sanger data together to obtain a consensus assembly [19]. It introduces a new term "super-reads" which it obtains from the error corrected original paired reads. Superreads contain all information as original reads and they are less in number. To create the super-reads, k-mers of the error corrected original reads are stored in a k-mer hash table. The hash table is used to extend each original read from both sides into a super-read. Every original read is in a super-read, many of them result in the same super-read, so there is no data loss and none of the unnecesary sequences are kept. Masurca removes duplication from the long read library by extracting maximal super-reads among super-reads which is not substring of another super-read. It then provides all available super-reads, linking mate pairs, long read mate pairs and other long reads as input to modified CABOG assembler [18]. Then it fills the gaps with the help of paired reads. CABOG [18] was revised to use 454 data, but it also accepts Illumina data to generate a hybrid assembly. MIRAest [68] can use Sanger, 454, Illumina, Ion Torrent and corrected PacBio data for hybrid assembly. It works on small genomes.

Some of the problems in *de novo* assembly can be ameliorated through using data generated by different sequencing platforms, where each technology has "strengths" that may be used to fix biases introduced by others. Other than hybrid assembly, there are also strategies in the literature to merge different assemblies using different data sources into a single coherent assembly (e.g. [71]).

With this study, we propose a new method to improve draft assemblies (i.e. produced using a single data source, and/or single algorithm) by incorporating data generated by different HTS technologies, and by applying novel correction methods.

To achieve better improvements, we exploit the advantages of both short but low-error-rate reads and long but erroneous reads. Our method differs from that of [71], in data types. [71] works on Illumina, 454 and ABI SOLID data, where we work on Illumina, 454 and Ion Torrent data. Also pre- and post-processing steps of the two methods differ. [71] at first assembles 454, Illumina and SOLID data separately with different assemblers and then assembles the resulting contig collection again with another assembler. We show that correcting the contigs built by assembling long reads through mapping short and high quality read contigs produces the best results, compared to the assemblies generated by algorithms that use hybrid data all at once. With this study, we also give a comparative evaluation of Ion Torrent and Roche-454 data in terms of their assembly performances. We also show that using three data types all together further improves the assembly quality. We compare our results with existing hybrid assemblers, CABOG and Masurca.

In Section 3.6 we give the properties of the datasets, the pre-processing operations to clean the data before assembly, the methods used to assemble and then correct the assembly, and finally the evaluation methods used for the comparison of the results. In Section 3.7 we present the results in different evaluation categories.

3.6. Methods

3.6.1. Data

We first cloned a part of human chromosome 13 into a bacterial artificial chromosome (BAC), and sequenced it separately using three different HTS platforms Illumina, Roche-454, and Ion Torrent platforms. The data properties are shown in Table 3.1. Illumina read lengths are 101bp, Ion-Torrent read lengths change from 5 to 201, and 454 from 40 to 1,027. Illumina has the highest mean base phred score quality and Ion-Torrent has the lowest. We also obtained a "gold standard" reference assembly for this BAC, first using GRCh37-guided assembly generated using Mira [72] with the Roche-454 reads, and then correcting with the Illumina reads [73]. Since Roche-454 and Ion Torrent platforms have similar sequencing biases (i.e. problematic homopoly-

mers), we separated this study into two different groups: Illumina & 454 and Illumina & Ion Torrent. We applied the same method on the two groups and evaluated them separately which gave us the opportunity to compare Roche-454 and Ion Torrent data. The overview of the assembly improvement method is shown in Figure 3.1. The explanations of the column headers in the figure are as follows: Technology: The name of the sequencing technology used to produce the reads. Length range: Minimum and maximum lengths of the generated reads. Mean length: The mean length among all reads. Mean base qual: The average phred score sequence quality of all reads. Calculated by summing up all phred scores of the bases in a read and dividing it to sequence length of the read, over all reads. Paired: Represents whether the sequencing is performed as paired-end or single-end.

Table 3.1. Properties of the data

Technology	**Length range**	**Mean length**	**Mean base qual (phred s.)**	**Paired**
Illumina	101bp (all reads have equal length)	101bp	38	paired
Roche-454	40bp-1027bp	650bp	28	single-end
Ion Torrent	5bp-201bp	127bp	24	single-end

3.6.2. Pre-processing

We applied the following pre-processing steps:

- First, we discard the reads that have low average quality value, which are the reads with phred score less than 17, i.e. $\geq$2% error rate (Figure 3.3).
- Then, we remove the reads with high N-density (with >10% of the read consisting of Ns) from consideration. Ns would destroy the assembly contiguity (Figure 3.4).
- Third, we trim the groups of bases at the beginning and/or at the end of the reads that seem to be non-uniform according to sequence base content (A,T,G,C). In Figure 3.2 we see that the first 4 bases and the last 30 bases show non-uniform (A,T,G,C) distribution. These regions would cause erroneous structures in the assembly.

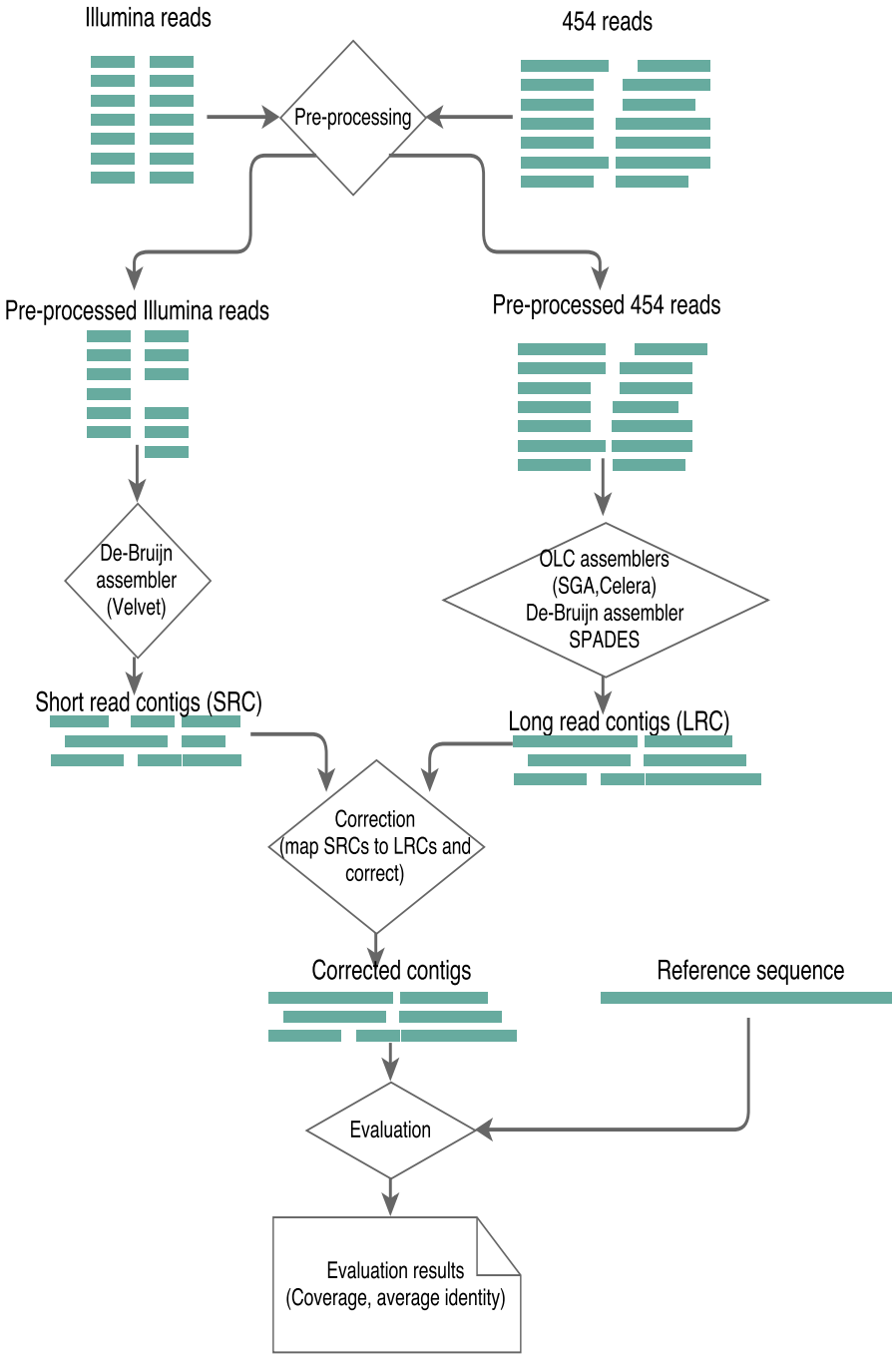

Figure 3.1. Overview of the assembly improvement method. Shown with Illumina and 454 data as an example. Same is valid for Illumina & Ion Torrent.

- Finally, we apply the pre-processing operations of each assembler we used.

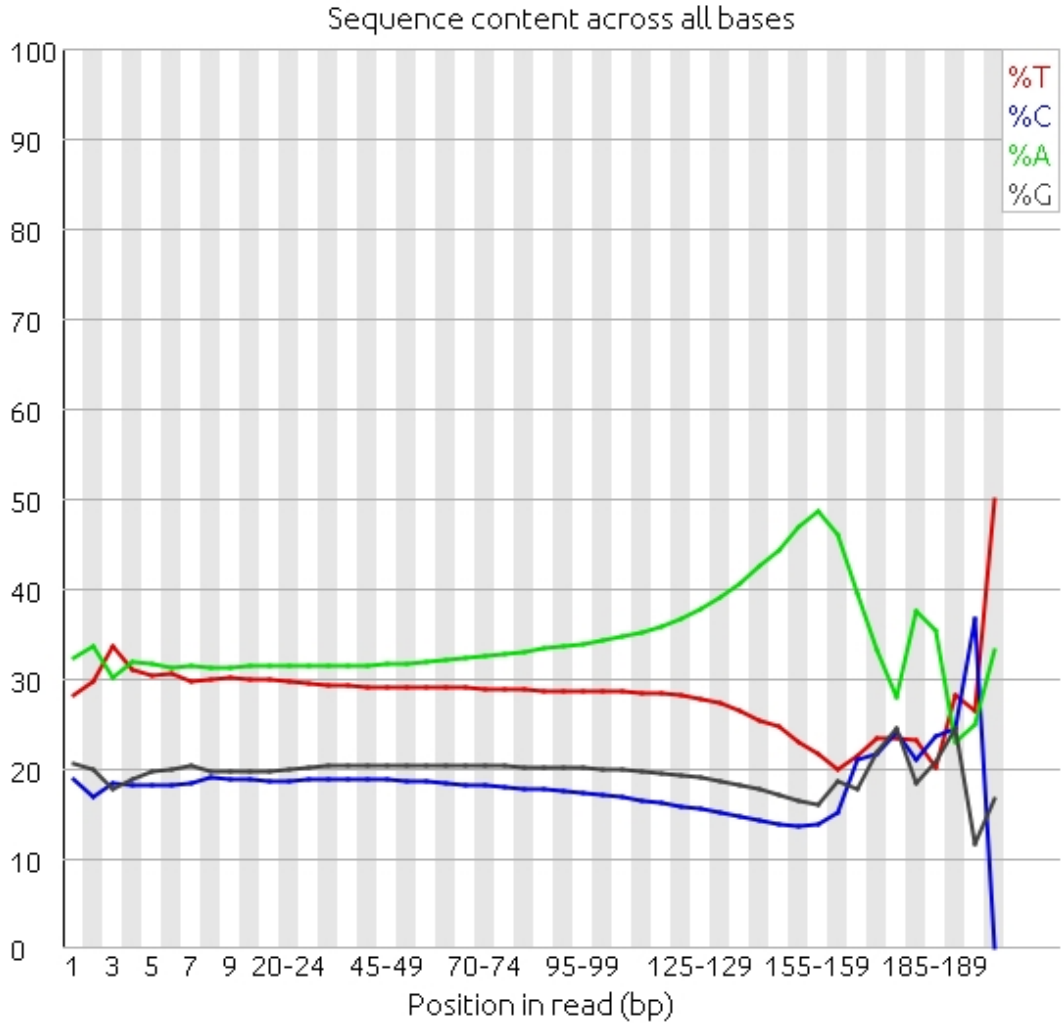

Figure 3.2. Non-uniform A,T,G,C regions of Ion Torrent reads. First 4 bases and the last 30 bases are trimmed in pre-processing.

3.6.3. Assembly

After the pre-processing step, we used several assembly tools which were suitable to the data types we have. We used Velvet [48], a de-Bruijn graph based assembler to assemble the Illumina reads. Considering the trimmed beginning and/or end parts of 101bp long paired-end reads from Illumina, and after testing k-mers 21 and 31, we decided to use k=51 for short read assembly. We ran Velvet with `shortPaired` mode with insert size 400bp, expected coverage 80, coverage cutoff 2, and minimum contig length 100. N50 value of the resulting short read contigs was 8,865 bp. To assemble the long read datasets (Roche-454 and Ion Torrent) we used two different

Input: Input read file in FASTQ format
Output: Filtered input file in FASTQ format
  for each read in input file **do**
    if *readlength* > 0 **then**
      $sumQS = 0$; // QS: Quality Score, SumQS: Sum of Quality Scores
      $i = 0$;
      while $i <$ read length **do**
        $sumQS \Leftarrow sumQS +$ (TO_DECIMAL(QS(read[i])))-33
      end while
      $sumQS \Leftarrow sumQS$/number of bases in the read
      if $sumQS > 17$ **then**
        keep the read
      end if
    end if
  end for
  return Output

Figure 3.3. Quality criteria algorithm

Input: Input read file in FASTQ format
Output: Filtered input file in FASTQ format
  for each read in input file **do**
    if *readlength* > 0 **then**
      if count(N) $<$ read length/10 **then**
        keep the read
      end if
    end if
  end for
  return Output

Figure 3.4. N-density criteria algorithm

OLC assemblers: Celera [17] and SGA [44]. We ran Celera assembler in `unmated` mode and with default parameters to assemble 454 and Ion Torrent reads. N50 value of the assembly obtained with 454 and Ion Torrent reads with Celera was 1,308 bp and 1,284 bp, respectively. We also used SGA assembler in unmated mode for the same datasets. We obtained N50 values of 505 bp and 117 bp for Roche-454 and Ion Torrent data, respectively. In addition, we also used a de-Bruijn graph based assembler, SPAdes [49], to assemble the long read data. Again, we applied default parameters. N50 values of the assemblies obtained with 454 and Ion Torrent reads with SPAdes were 212 bp and 259 bp, respectively.

We mapped all draft assemblies to the E. coli reference sequence using BLAST's [74] MegaBLAST [75] task to identify and discard E. coli contamination due to the cloning process. We discarded any contig that mapped to the E. coli reference sequence with sequence identity $\geq$95%. Finally, we obtained one short read, and three long read assemblies without any contamination.

3.6.4. Correction

In the correction phase, our purpose is to exploit the accuracy of the short read contigs (SRC) and the coverage of the long read contigs (LRC) to obtain a better assembly. Hence, we mapped all SRCs onto all LRCs of each group and corrected the LRCs according to the mapping results. First, we used BLAST's [74] MegaBLAST [75] mapping task to map the SRC onto the LRC. We then used an in-house C++ program to process the MegaBLAST mapping results. Since MegaBLAST may report multiple mapping locations due to repeats, we only accepted the "best" mapping locations. Reasoning from the fact that short reads show less sequencing errors, we preferred the sequence reported by the SRC over the LRC when there is a disagreement between the pair. By doing this, we patched the "less fragmented" long read assemblies. If there is an overlap between different SRC mappings at the same region on the LRC, the latter overwrites the first. Figure 3.5 shows a visual representation of the strategy on correcting the LRCs.

We describe our strategy in the following steps:

- If there is a mapping between a SRC and a LRC, and if the mapping does not start at the beginning of the LRC, add the unmapped prefix of the LRC.
- Next, if the mapping does not start at the beginning of the SRC (very rare situation), add the unmapped prefix of the SRC with lowercase (i.e. low confidence) letters.
- Over the mapping region between SRC and LRC, pick the SRC values.
- If the mapping does not end at the end of the SRC (rare), add the unmapped suffix of the SRC, again with lowercase letters. One may argue that it might disturb the continuity of the resulting contig. However, we observe such mapping properties very rarely. The reason for using lowercase letters is to keep track of the information that there is a disagreement between the SRC and LRC on these sections, so the basepair quality will be lower than other sections of the assembly.
- Finally, add the unmapped suffix of the LRC and obtain the corrected contig.

We repeated this process to correct each of the three long read assembly contig sets. We applied our correction strategy on each dataset multiple times until there is no improvement in the *Coverage* and *Average Identity* metrics.

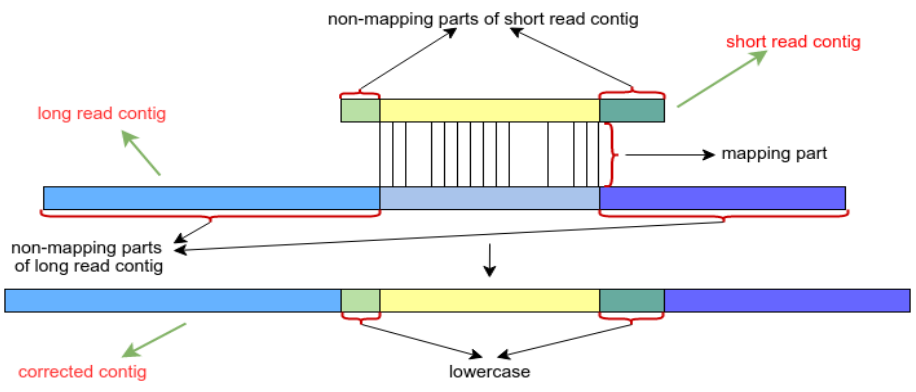

Figure 3.5. Correction method: Correct the long read contig according to the mapping information of the short read contig.

3.6.5. Correction of the Data from All Platforms

With our method, it is also possible to correct all three datasets in a sequential order with a pairwise strategy. Correction mechanism with three datasets is shown in Figure 3.6. Since our method is originally designed for two data types, we needed to do it sequentially. It corrects one data type's contigs with the other data type's contigs, so we needed to combine three types of contigs in order but the order is also important. As mentioned in Section 3.6.4 our method accepts that the short read data is more accurate than the long read data. If there is a map between the two, it replaces the values of the short read data with the values of the long read data. For that reason, while working with 3 data combination, we decided to use Velvet-Illumina contigs which are built by the highest accurate reads as the last short read corrector data.

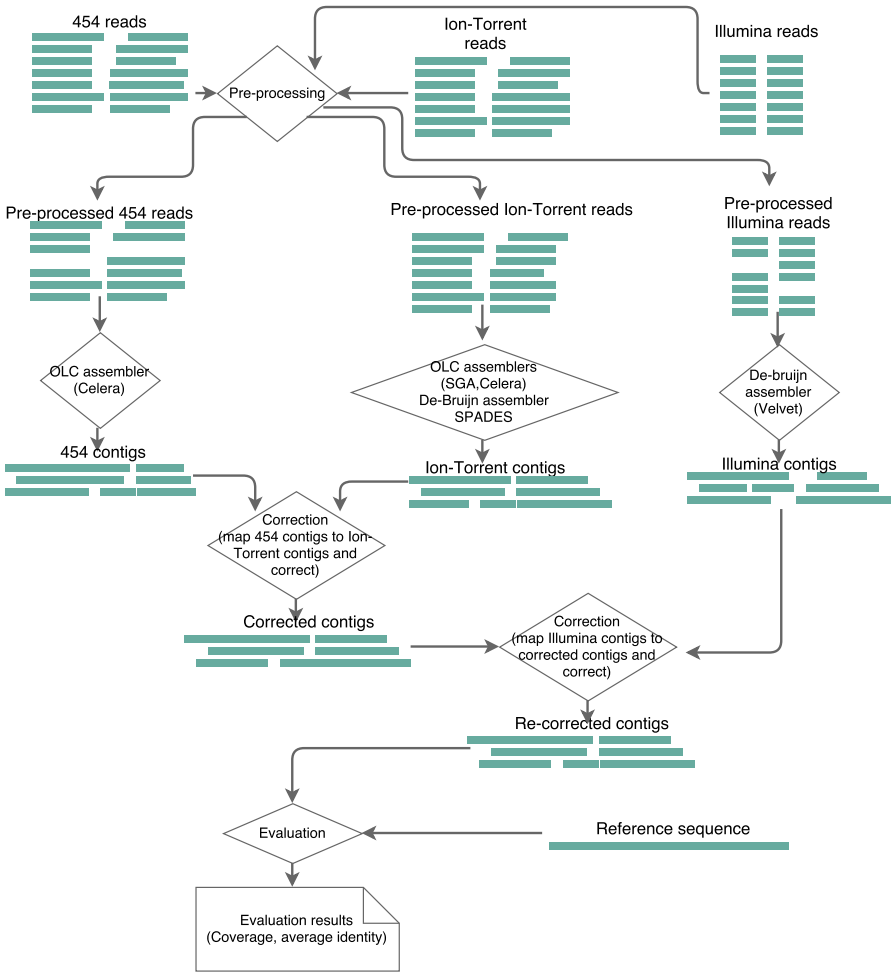

Figure 3.6. Correction method applied on three datasets together.

3.6.6. Evaluation

In order to evaluate and compare the resulting and corrected assemblies all-against-all, we mapped all of the assembly candidates, including primary assemblies and also final corrected assemblies to the "gold standard" BAC assembly. According to the alignment results, we calculated various statistics such as the number of mapped contigs, how many bases on the reference sequence were covered, how many gaps exist on the reference sequence, and the total gap length. We calculated metrics such as "Coverage" and "Average Identity" and compared the resulting assemblies with these metrics.

To calculate these statistics, we kept an array of arr_reference[0,0,0,...0], where length(arr_reference)= length(reference). We updated the contents of arr_reference according to the alignments. If there is a match at a location, we assigned the corresponding position in the array to "1"; if there is a mismatch at a location, we set it as "-1"; and if that location is not included in any alignment, we left it as "0" (which means a gap). We assumed deletions in the contig (query) as mismatches. We also calculated the number of insertions in the contig. Scanning the array and summing up the number of "1"s (matches), "-1"s (mismatches), "0"s (gaps) and "insertionInQuery", we obtained the number of matches, mismatches, gaps, and insertions in contig. Using these numbers, we calculated the Coverage (Equation 3.1) and Average Identity values (Figure 3.7).

$$\text{Coverage} = \left(\frac{\text{\# of covered bases}}{\text{length of the reference}} \right) \tag{3.1}$$

We also used two hybrid assemblers, Celera-CABOG [18] and Masurca [19], with Illumina & 454 and Illumina & Ion Torrent. These hybrid assemblers load all reads as input and assemble them with a hybrid method. We assembled the two datasets with these hybrid assemblers to compare our correction method with the results of them.

Table 3.2. Notations of Tables 3.3 and 3.4.

Notation	**Explanation**
Name	The name of the data group that constitutes the assembly
Length	Length of the assembly
# of Contigs	The number of contigs that belong to the resulting assembly
# of Mapped Contigs	The number of contigs that successfully mapped onto the reference sequence
# of Covered bases	The number of bases on the reference sequence that are covered by the assembly
Coverage	Percentage of covered reference
Avg. Identity	Percentage of the correctly predicted reference bases
# of Gaps	The number of gaps that cannot be covered on the reference genome
Size of Gaps	Total number of bases on the gaps.

Input: contigNum

Input: $match, mismatch, insertionInContig, contigLength$ for each contig

Output: $avgIdentity$ //$avgIdentity$: average Identity

$i \leftarrow 1$

for $i \leq$ contigNum **do**

$alignmentLen_i \leftarrow match_i + mismatch_i + insertionInContig_i$

$identity_i \leftarrow \frac{match_i}{alignmentLen_i}$

end for

$avgIdentity \leftarrow \frac{\sum_{i=1}^{contigNum} identity_i \times contigLen_i}{\sum_{i=1}^{contigNum} contigLen_i}$

return Output

Figure 3.7. Average identity algorithm

Table 3.3. Results of assembly correction method on BAC data.

Name	Length	# of Contigs	# of Mapped Contigs	# of Covered bases	Coverage	Avg. Identity	# of Gaps	Size of Gaps
Velvet								
Ill. Velvet	197,040	455	437	175,172	0.99055	0.97523	39	1,671
Celera								
454 Celera	908,008	735	735	172,563	0.97580	0.92599	18	4,280
Ion Celera	39,347	27	27	47,638	0.26938	0.96932	47	129,205
Corrected Celera								
Ill-454 Celera	371,065	250	250	176,071	0.995635	0.944558	5	772
Ill-454 Celera^2[*]	365,802	245	245	176,343	0.9971	0.9455	4	500
Ill-Ion Celera	93,909	30	28	81,819	0.46267	0.96327	36	95,024
Ill-Ion Celera^2	145,262	30	28	91,962	0.52002	0.97412	33	84,881
Ill-Ion Celera^3	216,167	30	28	99,645	0.56347	0.98066	34	77,198
SGA								
454 SGA	62,909,254	108,095	101,514	176,546	0.99832	0.97439	1	297
Ion SGA	842,997	6,417	6,122	153,092	0.86569	0.99124	197	23.751
Corrected SGA								
Ill-454 SGA	295,009	335	335	176,757	0.99951	0.96823	5	86
Ill-Ion SGA	197,509	291	291	175,052	0.98987	0.97501	45	1,791
Ill-Ion SGA^2	203,064	291	291	175,676	0.99340	0.97413	34	1,167
SPAdes								
454 SPAdes	12,307,761	49,824	49,691	176,843	1.0	0.98053	0	0
Ion SPAdes	176,561	110	107	167,890	0.94937	0.92909	9	8,953
Corrected SPAdes								
Ill-454 SPAdes	290,702	298	298	176,454	0.99780	0.96538	5	389
Ill-Ion SPAdes	198,665	52	52	171,977	0.97248	0.94215	4	4,866
Ill-Ion SPAdes^2	200,307	52	52	172,101	0.97319	0.94230	2	4,742
Masurca								
Ill-454 Masurca	380	1	0	0	0	0	0	0
Ill-Ion Masurca	2,640	8	8	1,952	0.01104	0.98223	9	174,891
Celera-CABOG								
Ill-454 Celera	1,101,716	891	891	174,330	0.98579	0.92452	12	2,513
Ill-Ion Celera	0	0	0	0	0.0	0.0	0	0.0

Reference Length is 176,843. ^{2} represents the results of the second cycle of correction, ^{3} represents the third cycle of correction.

Table 3.4. Results with combination of 3 data types

Name	**Length**	**# of Contigs**	**# of Mapped Contigs**	**# of Covered bases**	**Coverage**	**Avg. Identity**	**# of Gaps**	**Size of Gaps**
Corrected Ion Celera								
454-Ion Celera	500,251	27	27	149,021	0.84267	0.94515	63	27,822
Ill-"454-Ion Celera"	570,865	27	27	170,348	0.96327	0.95503	16	6,495
Ill-"454-Ion Celera"[2]	575,726	27	27	172,516	0.97553	0.95541	12	4,327
Ill-"454-Ion Celera"[3]	578,727	27	27	174,535	0.98694	0.95555	10	2,308
Corrected Ion SPAdes								
454-Ion SPAdes	11,224,602	60	60	176,540	0.99828	0.97334	6	303
Ill-"454-Ion SPAdes"	9,543,712	45	45	176,712	0.99925	0.97347	1	131
Corrected Ion SGA								
"Ill-454"-Ion SGA	281,155	212	212	176,769	0.99958	0.96562	4	74
Masurca(all)								
Ill-454-Ion Masurca	3,398	7	5	1,477	0.00835	0.99363	5	175366
Celera-CABOG(all)								
Ill-454-Ion Celera	575,642	485	485	164,621	0.93088	0.94664	39	12,222

Reference Length is 176,843. [2] represents the results of the second cycle of correction. [3] represents the third cycle.

3.7. Results

We present the results in Table 3.3, and interpret them in different point of views. The results show that our novel correction method generates new assemblies with better coverage and better identity compared to the existing hybrid assemblers. In addition, according to the results, 454 assembly is more complete and accurate than Ion-Torrent assembly. Although Ion-Torrent assembly is improved with the correction method, this property is also transferred to the final results, corrected 454 assembly is better than corrected Ion-Torrent assembly. Assembling Illumina reads with Velvet assembler seems to be a good choice as well as assembling 454 reads with Celera and assembling Ion-Torrent reads with SPAdes. Correcting separately obtained contigs using each other with our new method gives more accurate and complete final assembly than assembling different kinds of data together with the existing hybrid assemblers. Best results are obtained with correcting all three data sequentially with our method.

3.7.1. 454 vs. Ion Torrent

Ion Torrent reads are shorter than 454 reads and they have less mean base quality (Table 3.1). So, we did not expect to have better assembly with Ion Torrent reads than 454 reads. The results in Table 3.3 agree with our expectations. In Table 3.3, we see that the assembly of 454 reads performs better on evaluation metrics than Ion Torrent reads with all kind of assemblers. The assembly of Ion Torrent reads with Celera assembler has very low coverage value: 26.94%. Although 454 and Ion Torrent reads have similar error types at the homopolymer regions Celera had very low coverage on Ion-Torrent data. The reason for the low coverage might be because Celera assembler is not designed for Ion Torrent read type (shorter reads with lower quality). SGA assembly with Ion Torrent reads performs better on Coverage (86.57%) but it cannot reach to the Coverage of SGA assembly with 454 reads (99.83%). The assembly of Ion Torrent reads has the highest coverage with SPAdes assembler (94.94%). Correction of the Ion Torrent contigs improves the assembly quality but even after correction phase Ion Torrent corrected assembly cannot reach the results of 454 corrected assembly.

3.7.2. Assemblers

Table 3.3 shows that the assembly obtained by Velvet with only short Illumina reads showed good coverage (99.05%) and average identity rates (97.52%). The number of contigs obtained with Velvet assembly is 455, of which 437 map to the reference. There are 39 gaps and the total size of the gaps is 1,671 bp. Our aim was to increase the coverage, improve the average identity, decrease the number of contigs and gaps, and shrink the lengths of the gaps.

Since we observed that 454 reads resulted in better assembly than Ion Torrent reads as stated in Section 3.7.1, we compared different assemblers using 454 contigs. The assembly of Celera with the 454 long reads has 97.58% coverage and 92.59% average identity, which are lower than Illumina-Velvet values. Number of contigs (735) is reasonable but number of gaps and total gap length are high (18 and 4,280 bp, respectively). SGA assembly using 454 reads has very high coverage (99.83%) and identity (97.43%). It has just one gap with size 297 bp, but the number of contigs is also very high (101,514), which is not welcomed. SPAdes-454 assembly also had a large number of contigs (49,824) which completely cover the reference sequence with 98.05% average identity. SPAdes assembly resulted in lower number of contigs and had higher coverage and average identity than SGA. If we evaluate the results according to the number of contigs, Celera-454 results seem more reasonable than SGA or SPAdes results, since it returned a reasonable number of contigs even with low coverage and average identity.

3.7.3. Correction

We observed that the correction method improved both 454 and Ion Torrent based assemblies generated with all assemblers we tested (Table 3.3). In the remainder of the paper, we only mention the 454-based assemblies for simplicity.

When we applied our correction method on Celera-454 assembly using the Velvet-Illumina assembly, we achieved better coverage and average identity rates: the coverage

of 454 assembly increases up to 99.56% and the average identity rate increases up to 94.45% on the first correction cycle. The second correction cycle increases the coverage and average identity rates to 99.71% and 94.55%, respectively, and the correction converges. The number of contigs decrease to 245 from 735, and the number of gaps decrease down to 4 (500 bp) from 18 (4,280 bp). Since the third correction cycle does not give better results it is not shown in Table 3.3.

Our correction method increased the coverage of SGA-454 assembly up to 99.99% from 99.82% but with less average identity and with more gaps although the total length of the gaps is decreased. Correction using the short read assembly decreased the number of contigs down to a reasonable number (335). Corrected SGA assembly has the largest coverage rate among all, and also with more identity than Velvet-Illumina assembly.

The number of contigs in SPAdes assembly also decreased to 298 from 49,691 using our correction method. With the decrease in number of contigs, the coverage also decreased (99.78%) as well as the average identity (96.53%). The number of gaps increased to 5 from 0 with a total size of 389.

In summary, we obtained substantial assembly correction in draft assemblies by using advantages of different technologies.

3.7.4. Hybrid Assemblers

We also compared the results of two hybrid assemblers on our multiple type of data. We used Masurca and Celera-CABOG with default parameters given two groups of hybrid data as input: Illumina & 454 and Illumina & Ion Torrent. Hybrid assemblers Masurca and CABOG did not show good assembly rates. We obtained zero coverage with 454 and Illumina reads using Masurca. The only contig left after the contamination removal did not map to the reference sequence. We also observed very low coverage (1.10%) with 98.22% average identity with Ion Torrent & Illumina reads. Therefore, we conclude that Masurca did not work very well in our case with our data

types.

Similarly, we obtaizero coverage with Ion Torrent & Illumina using CABOG. All of the resulting contigs obtained from the assembly were removed as contamination. However, CABOG performed substantially better with Illumina & 454, and generated assembly with 98.58% coverage and 92.45% average identity. The assembly composed of 891 contigs and 12 gaps with a total gap length of 2,513 bp. Still, the performance of CABOG was not better than the corrected assembly results described above.

3.7.5. Combination of the Data from All Platforms

We combined data from all three platforms to generate a new assembly in order to see if we have better coverage or accuracy on the results. The results are presented in Table 3.4. In the table one can see that Celera-454 contigs increase the coverage rate of Celera-Ion Torrent contigs (from 26.93% to 84.26%) although the average identity rate decreases from 96.93% to 94.51%. Correcting the resulting contigs with Velvet-Illumina contigs increases the coverage (96.32%) and average identity rates (95.50%) even higher. The coverage and average identity rates are improved on the second and third cycles too. Correcting Ion SPAdes, with 454 SPAdes gives higher coverage (99.82%) and average identity (97.33%) rates than correcting them with only Velvet Illumina contigs (97.24% and 94.21% respectively). After using Velvet Illumina contigs for the last correction, the results are improved approximately by 0.1% and 0.01% respectively. Correcting Ion SGA contigs with 454 SGA contigs was not possible because of memory limitations of BLAST mapping with such huge data. Instead, we used corrected version of "454 SGA contigs with Illumina Velvet contigs" to correct the Ion SGA contigs. The coverage is higher than both Ill-Ion SGA and Ill-454 SGA, average identity is lower than Ill-454 SGA.

We also used the hybrid assemblers Masurca and CABOG with default parameters with the combination of three data. Masurca resulted in very low coverage 0.8% as it did before with the dual combinations. CABOG resulted in lower coverage and higher average identity compared to Ill-454 combination and higher in both compared

to Ill-Ion Torrent combination. Hybrid assembler still did not result in as high coverage and average identity as obtained with the correction method.

We note that exploiting all the data gives us more accurate results especially when we are using a diverse data which has different strengths and weaknesses. However, one must be careful about the weaknesses and strengths of the data and where and in which order to use each of them.

3.8. Discussion

In this study, we presented a novel method to improve draft assemblies by correcting high contiguity assemblies using the relatively more fragmented contigs obtained using high quality short reads. Assembling short and long reads separately using both de-Bruijn and OLC graph based assemblers according to data types and then using correction methods gives better results than using only hybrid assemblers. Using three data types together for correction or as the input of the hybrid assemblers rather than using only two of them gives more accurate results.

However, the need to develop new methods that exploit different data properties of different HTS technologies, such as short/long reads or high/low quality of reads, remains. In this manner, as future work, our correction algorithm can be improved by exploiting the paired-end information of the short, high quality reads after the correction phase to close the gaps between corrected contigs.

We also note that, the biggest challenge of combining data from multiple platforms is the cost of the sequencing platforms. Sequencing with Illumina platform is cheap compared to other platforms but sequencing the same DNA with different platforms highly increases the cost. Therefore, it cannot be used as part of the routine analysis especially for high coverage whole genome sequencing of large genomes such as human genome.

4. DISCOVERY AND GENOTYPING OF NOVEL SEQUENCE INSERTIONS IN MULTIPLE INDIVIDUALS

Genomic structural variations (SVs) are broadly defined as alterations that affect more than 50 base pairs (bp) of DNA [13], and they have major impact on both evolution and human disease [13, 76]. Such alterations may be in various forms including deletions, insertions, inversions, duplications, and retrotranspositions [13]. Thanks to the wide availability and cost efficiency of high throughput sequencing (HTS), we now have the ability to characterize SVs in the genomes of many individuals, as exemplified by large-scale projects such as the 1000 Genomes Project [77, 78]. Accurate characterization of SVs required the development of many novel algorithms [13, 79] that are benchmarked within the 1000 Genomes and the Genome in a Bottle [11] projects.

Novel sequence insertions, or alternatively, "deletions from the reference", are genomic segments that are not represented in the reference genome assembly [15]. Similar to "deletions from the sequenced sample", they may harbor sequences of functional importance such as coding exons or regulatory elements [15], which underline the importance of their accurate characterization. The non-reference sequences identified in various genome studies are thus "added" to the reference genome as additional sequence. However, due to the complexity of these new sequences and their polymorphism in different populations, there is now a push towards building graph-based representations [80, 81].

Although several forms of SVs such as deletions, tandem duplications and mobile element insertions are investigated to a certain extent [4, 13, 14], characterization of novel sequence insertions longer than read lengths is still lagging. This is mainly because long sequence insertions can be discovered only through sequence assembly, which is computationally challenging and may lead to incorrect or fragmented sequence reconstructions due to common repeats that may lie within or close to such insertions [15, 82].

Pindel is a tool that detects large deletions and medium sized insertions from paired-end short reads [83]. It depends on split read information to detect the breakpoint. It splits the unmapped ends of the reads into three parts and remaps them separately to the reference genome. If any of them maps to the reference sequence, it is defined as the anchor location and it shows from where to split the whole read. It then applies pattern growth to find the minimum and maximum unique substrings from the $3^{'}$ and also from the $5^{'}$ of unmapped read. It checks both ends if they are identical to each other and reports the middle sequence as the insertion if its support is greater than 2. Pindel can only detect insertions shorter than the read length. Another tool to detect structural variants is Cortex [84]. It aims to improve the accuracy in complex regions by using colored de-Bruijn graphs to detect and genotype the insertions in one or many individuals. A recent study found that it has high computational requirements [1].

Aside from computationally intensive assembly-based algorithms, only a handful of mapping and local assembly based methods for novel sequence insertion discovery are currently available. The first of such algorithms is NovelSeq [82] which is developed to find insertions >200 bp using paired-end whole-genome Illumina sequence data.

Briefly, NovelSeq extracts the unmapped reads (one-end-anchors (OEA) and both-end-unmapped reads (orphan)) from the alignment output and maps them back to the reference with multi-mapping. It assembles the orphan reads with a *de-novo* assembler such as ABySS [47] or Velvet [48]. It clusters the OEAs and divides them as OEA+ for the OEAs whose mates are aligned at the beginning of the breakpoint and OEA- for the OEAs whose mates are aligned at the end of the breakpoint. Clusters of OEA contigs are also constructed according to a distance rule. The beginning of the cluster location should not exceed the end of the cluster location more than a distance threshold (mean fragment size + four standard deviations). Each OEA+ and OEA- cluster are assembled locally. The OEA contig clusters and the orphan contig sets are merged together with a bi-partite graph matching algorithm and the pairs of OEAs and orphan contigs are obtained. NovelSeq identifies both the content and the approximate location of the insertion. However, NovelSeq was designed to analyze

one genome at very high sequence coverage. It could find insertions of length up to a couple of kilobase pairs and it does not provide the exact content of the insertion, the exact breakpoint location and the genotyping information. Similarly, MindTheGap [2] was developed for finding insertions in a single sequenced genome. It uses an efficient de-Bruijn graph based method. It first constructs the de-Bruijn graph with the reads with Minia [51]. It then detects the insertion breakpoints on the reference genome and finally it locally assembles the inserted sequences. It uses the same constructed graph for both detection and assembly. The genotyping is also done at the breakpoint detection stage.

Another algorithm, Swan [85] can *only* find breakpoints of long insertions using "soft-clipped" reads. However, it does not report the content of the insertion. It collects the estimation of library-specific parameters first, then it scans the whole genome alignment file to find possible variant locations with two different methods: (i) a likelihood ratio-based method and (ii) split-read remapping method. Next, it merges the outputs obtained from two methods.

BASIL and ANISE [3] are also designed for detecting insertion breakpoints and assembling the insertions, respectively. BASIL implements an efficient sliding window for clustering one-end anchored reads similar to NovelSeq [82] and ANISE assembles the novel insertions with an overlap-layout-consensus graph based assembler approach which is robust to repeated copies, by combining ideas from [17, 18, 68].

A more recent algorithm, PopIns [1] follows a similar approach and also incorporates the split-read sequence signature [13] to discover the sequence insertions in one sample or a large cohort of samples. It first assembles the unmapped reads of each individual separately. Then it merges the generated contigs into a higher quality multi-individual contig set. Next, it finds the locations of these contigs on the reference genome through paired-end and split-read read information. Lastly, it assigns the genotypes (homozygous, i.e. two copies, heterozygous, i.e. one copy, and no insertion) of each call for each individual by mapping the raw reads of each individual separately.

In this study, we present a novel algorithm to efficiently and accurately discover novel sequence insertions in single or multiple genomes sequenced with the Illumina technology and based on this algorithm we developed a new tool called Pamir. Pamir provides exact breakpoint positions, sequence contents, and genotypes of novel sequence insertions. In order to discover insertions, Pamir matches one-end anchored sequences to *de novo* assemblies of unmapped reads and generates a strand aware local assembly. It outperforms both MindTheGap [2] and PopIns [1] when a high coverage simulated single genome is used. Additionally, using simulated low coverage data (5 samples at 10X coverage each) we demonstrate that Pamir has better precision and recall rates than PopIns, which is the only other insertion characterization tool that can use multiple genomes. We describe our method in Section 4.1 and we present our results on both simulated data and real data in Section 4.3.

4.1. Methods

We developed Pamir to characterize novel sequence insertions using paired-end whole genome sequencing (WGS) data generated by the Illumina platform. Pamir is based on the observation that structural events such as "novel sequence insertion" leave a group of one-end anchors around their breakpoint location when aligning the donor sequences to the reference genome [15, 82, 86]. One-end anchor is one of the signatures of the structural variations in the paired-end mapping, where one-end of the pairs is mapped while the other is unmapped. By definition, a novel sequence does not exist on the reference genome, thus it also causes some read pairs to remain unmapped, which are called *orphan* reads, i.e. when none of the ends are mapped. Orphan reads are seen if the inserted sequence is at least as long as the paired-end fragment size.

Figure 4.1 depicts the mapping information in the vicinity of the hypothetical novel insertion. The explanations of the terms used in the figure are as follows: *Concordant read*: both ends map in correct orientation and within expected insert size. *OEA read*: one-end anchored, only one end maps to the reference. *Split read* is an OEA read whose unmapped end crosses the breakpoint and generates split mapping. *Orphan read*: none of the ends map to the reference. Pamir uses both types of reads to

characterize the novel sequence contents and their insertion breakpoints. First, it starts with generating a *de novo* assembly of the orphan reads to obtain *orphan contigs*. Next, Pamir clusters the OEA read pairs based on their mapping locations on the reference genome. It then remaps the OEA reads to orphan contigs to match the orphan contigs with OEA clusters. It outputs the putative novel insertion by assembling the updated cluster and re-aligning the generated contig to the respective reference region (Figures 4.2, 4.3). Finally it applies filtering procedure to eliminate the low quality insertions and finds the genotypes of the insertions. Genotype is the "zygosity" of the event if it is seen as one copy (heterozygous), two copies (homozygous) or not seen (no insertion). In this section, we provide a detailed description of the Pamir algorithm: Section 4.1.1 explains the pre-processing steps, such as unmapped read extraction, remapping OEAs, orphan assembly and matching between OEAs and orphan contigs. Section 4.1.2 gives details about cluster formation. Section 4.1.3 describes the discovery of the novel sequence insertions. Section 4.1.4 explains the filtering and genotyping phases.

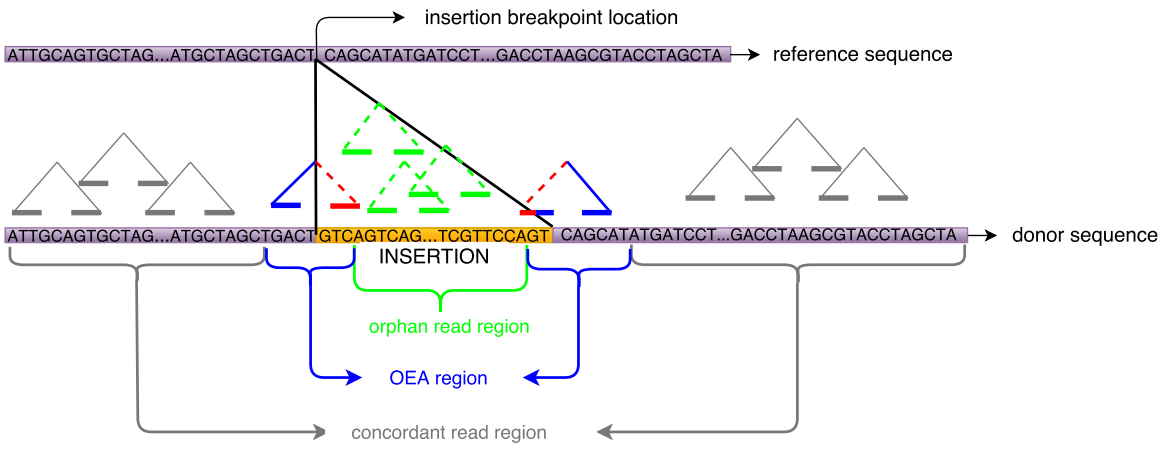

Figure 4.1. Classification of donor sequence regions in terms of read mappings.

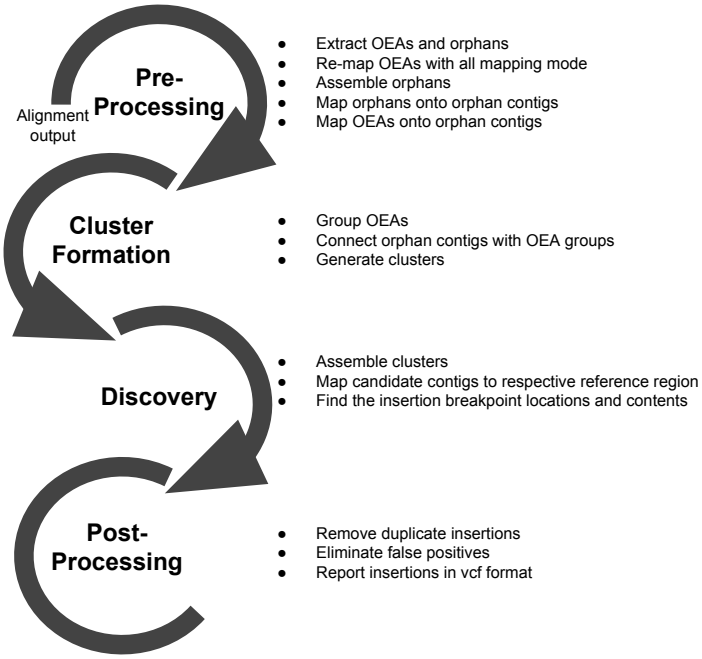

Figure 4.2. Overview of Pamir.

4.1.1. Pre-processing

Pamir accepts both raw reads in FASTQ format or aligned reads as BAM files as input. If raw reads are provided, Pamir first maps them to the reference genome using mrsFAST-Ultra [87, 88] in best mapping mode. Pamir skips the mapping step if the read alignment is provided, i.e. BAM file. Next, Pamir extracts OEA and orphan reads using the alignment results. OEA and orphan reads are shown in Figure 4.1. Pamir then remaps the OEA reads using mrsFAST-Ultra in multi-mapping mode since the breakpoints of a sequence insertion may lie within repeats, which causes mapping ambiguity [39, 89] (Figure 4.3A). We track many possible locations of the OEAs with multi-mapping by allowing a generous enough number of mappings which can also be decided by the user. Using multi-mapping locations may introduce false positives in repeat regions, which we eliminate in a post-processing step. We sort the multi-mapping alignment output according to the OEA locations on the reference. The OEAs are needed as sorted in the clustering phase.

Pamir assembles the orphan reads using Velvet [48] with the k-mer length set to 31bp, although any other assembler may also be used for this step (Figure 4.3B).

We should remark that parameter setting during the assembly stage based on the data type has significant role on obtaining complete orphan contigs which construct the main body of the long insertions. If orphan assembly fails due to some reasons, such as the wrong parameter setting, low coverage, and high error rate, we do not obtain any orphan contig. Therefore left and right flanks constructed by only OEAs will have a gap in between which is supposed to be filled by the orphan contig. In such a situation we cannot report the long insertion. In order to decide on the assembler, we tried SGA [44] with overlap length 71, 51, and 31, and Velvet and Minia [51] with k-mer length 31bp on simulation data experiments. We observed that using SGA with k-mer=51 results in approximately $\sim\%2-\%3$ better recall rates ($\sim\%92-\%98$) on each dataset compared to Velvet, but it causes $\sim\%28$ lower recall rate ($\sim\%53$) on HiSeq2000, 100bp noisy data. Minia was doing slightly better than Velvet on each high coverage dataset but it had even less recall rate than SGA on low coverage simulated data. Therefore, we decided using Velvet with k-mer=31bp which resulted good enough recall rates ($\sim\%81-\%96$) on all datasets, which is better comparing to other competitor tools on all datasets. The default assembler is Velvet, but Velvet might have some high memory usage problems on very high coverage data, Minia has very efficient memory usage. To allow the user making changes on the assembler according to their data type, we kept assembler type optional. Two other options allowed by Pamir for now are: (i) SGA [44], an overlap layout consensus (OLC) graph based assembler with 51bp overlap length, and (ii) Minia [51] which is a *de-Bruijn* graph based assembler with the k-mer length set to 31bp. After the assembly, we subject the contigs to a contaminant filter by querying the `nt/nr` database with BLAST [90]. Before using orphan contigs in further steps, we remove those contigs that map to vector and/or bacterial sequences and other known contaminants. Support value of the contig (i.e the number of reads constructing the contig) is not accessible, since redundant reads are discarded during the assembly but we calculate the support value of the insertion at the last (genotyping) stage. We then map the unmapped end of OEA read pairs to the orphan contigs using mrsFAST-Ultra in the multi-mapping mode to match the OEAs to the corresponding orphan contigs. In this way, the OEA-to-orphan remapping stage allows an OEA to be aligned to more than one orphan contig (Figure 4.3C). In order to avoid missing

any associations between split reads (Figure 4.1) and orphan contigs, we divide the unmapped OEAs into half (i.e. balanced splits [91]) and remap the balanced splits to the orphan contigs.

In summary, the pre-processing step generates four types of information required to discover a novel sequence insertion event:

(i) the mapping information of the OEA mapped reads;
(ii) unmapped OEA sequences;
(iii) orphan contigs; and
(iv) pairwise association between unmapped OEA reads and orphan contigs

which are going to be used in the following steps.

4.1.2. Cluster Formation

Pamir clusters the unmapped OEAs based on the mapping locations of their mapped end to detect potential insertion breakpoints. It also keeps a flag for the possible strand of the unmapped OEA sequence either as forward or reverse complemented according to its mate's mapping strand. It then employs an iterative greedy strategy, which "anchors" the first cluster with the "leftmost" mapping locus a of an OEA on the genome. Next, it extends the cluster to include any other OEA mappings overlapping with the interval $\left[a, a+2L\right]$ where L is the fragment size. Fragment size can be described as follows: Let L be the fragment size of paired-end reads which can be estimated from concordant mappings. For an insertion in breakpoint p, most of its OEA anchors should be mapped within $\left[p-L, p+L\right]$, which spans a $2\times L$ interval on the reference genome. Once all such OEA mappings are added to the existing cluster, the iterative strategy then greedily anchors the next cluster with the first OEA mapping that is not included in the previous cluster. An example of two clusters is shown in Figure 4.4. In the figure, the clusters consist of only unmapped OEAs (red reads) and an assembly of orphans (green reads). In the first cluster insertion is longer than the fragment size, therefore an orphan contig will be included in the cluster too. In the

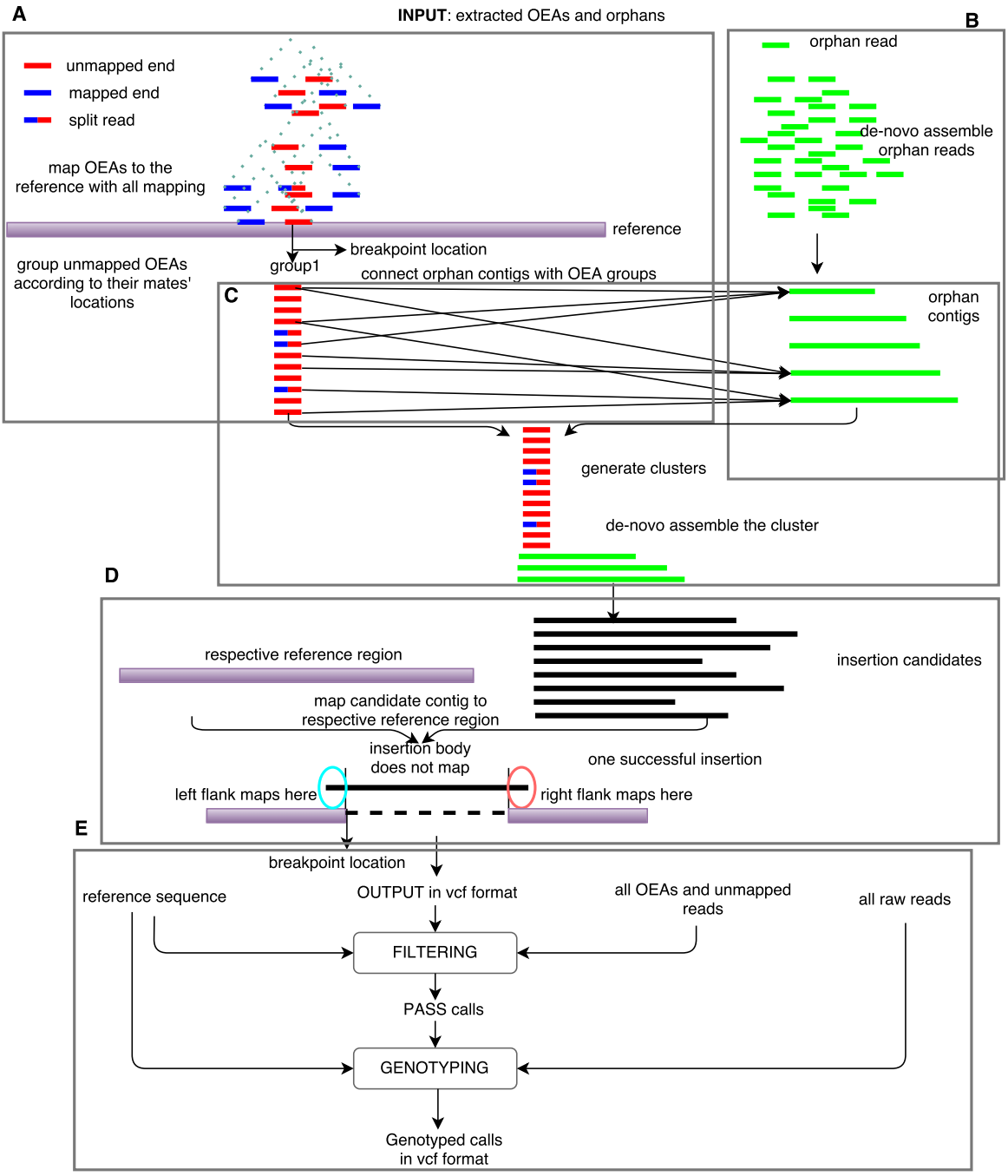

Figure 4.3. General overview of Pamir.

second cluster, insertion is shorter than the fragment size, so the cluster will include only OEAs.

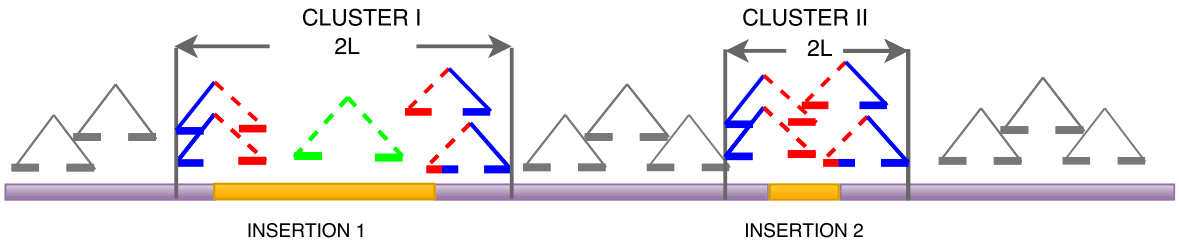

Figure 4.4. Example clusters of Pamir for two insertions.

Note that in this strategy each OEA mapping can only be part of a single cluster. However, a single read pair may generate multiple OEA mappings (and thus belong to multiple OEA clusters) due to the use of multi-mapping strategy. Since both ends of the orphan reads do not map to the reference genome, we have no locus information for orphans, as well as orphan contigs. The "OEA-to-orphan contig" mapping information reveals if they are part of the same cluster (same structural variation). After the first clustering pass is completed, Pamir adds the unmapped OEA mates of the reads and their associated orphan contigs into each cluster (Figure 4.3C).

To find the associated orphan contigs, the "OEA-to-orphan contig" mapping information generated in the pre-processing step is used. A contig is added to a cluster if (i) the cluster contains OEAs that map to the both ends of the orphan contig; or (ii) at least 30% of the OEAs in the cluster map only to either end of the contig. We allow the second condition to avoid missing any partially assembled orphan contigs. In order to decide on the orphan contig's strand (forward or reverse complement), we check OEA reads' mapping strands on the reference sequence and also on the orphan contig. We add the contig forwardly if majority of the OEAs' mapping strands on the reference and the orphan contig agree with each other. The contig is added as reverse complemented if the aggregation is on disagreement with each other.

In summary, each cluster generated in this step contains the following information:

(i) the number of the OEA reads and their associated contigs;
(ii) the leftmost OEA mapping location;
(iii) the rightmost OEA mapping location;
(iv) unmapped OEA read information (see below); and
(v) contigs associated with unmapped OEA reads.

For each unmapped end of an OEA read pair, the following information is kept in the cluster:

(i) read name;
(ii) strand (based on its corresponding mapped mate); and
(iii) read sequence.

The clusters with less than three members are not considered for insertion discovery.

4.1.3. Insertion Discovery

4.1.3.1. Candidate Insertion Contigs. Pamir generates a new assembly for each cluster to compute the putative insertion. A putative insertion consists of its main body which constitutes the insertion (Figure 4.3C) and also both left and right flanking regions that overlap with the reference genome. The resulting cluster-aware assembly represents a potential novel insertion sequence. We keep the OEAs in the cluster according to their strand information and the orphan contigs' strand is also decided on the OEA-orphan contig matching stage, so the cluster is aware of the potential insertion's strand. The assembler assembles the cluster according to the given strand; it considers only forward assembly not reverse complement. This gives us the cluster-aware assembly.

We assemble the reads and contigs in each cluster using an efficient in-house overlap-layout-consensus (OLC) assembler. We found most of the available off-the-shelf assemblers to be too slow for this task, especially because the total number of

clusters is measured in millions. Additionally, existing tools cannot be modified to consider strand information that can be inferred from the mapping information while our in-house assembler is strand-specific. Furthermore, the use of naïve greedy strategy for assembly is not suitable for our goal because such a method cannot obtain optimal contigs necessary for accurate insertion detection.

The objective of the in-house assembler is to construct a contig that maximizes the total sum of overlaps between the reads. This problem can be optimally solved by modeling it as an instance of *maximum weighted path* problem in a directed graph G as follows. Let each vertex v represent a read in the cluster. Two vertices m and v are connected with a directed edge $e_{m,v}$ of weight $w_{m,v}$ if the maximum prefix-suffix overlap between the reads represented by those vertices is of length $w_{m,v}$. We can optimally calculate the maximum weighted path via a dynamic programming formulation as follows.

Suppose that there exists some ordering $<_v$ of the vertices of G, where $\text{parent}(v) <_v v$ always holds for any vertex v and its parent, $\text{parent}(v)$. Furthermore, let r be the root of the graph G (as long as $<_v$ exists, the root can be selected as the smallest vertex with respect to $<_v$). We can calculate the value of maximum path from the root r to any vertex v, denoted as $f(v)$, by the following recurrence:

$$f(v) = \max_{\text{parent}(v)} \left\{ f(\text{parent}(v)) + w_{\text{parent}(v),v} \right\} \tag{4.1}$$

assuming that initially $f(r) = 0$ for any root r (i.e. vertex with no incoming edges). The equation 4.1 can be implemented in iterative fashion by iterating over vertices v in order $<_v$. This dynamic programming formulation has the complexity of $O(|R| + |E|)$, where $|R|$ denotes the number of reads in the given cluster, and $|E|$ is the total number of edges in G. This recurrence will always produce the optimal solution as long as there exists ordering $<_v$ with the above-mentioned properties. The most natural choice for $<_v$ is topological ordering of G, which maintains the necessary invariant $\text{parent}(v) <_v v$. Topological ordering can be efficiently calculated in $O(|R| + |E|)$ via Kahn's algorithm [92].

However, both topological sorting and Equation 4.1 require acyclic G, which might not be always true, especially if the target region contains some repeat. In that case, maximum weighted path problem is NP-hard, which can be easily shown by reducing the longest path problem in a graph to the maximum weighted path problem. If cycles are present, we remove any cycle from G in a greedy fashion by iteratively removing cycle edges whose endpoint is the vertex with the largest indegree in G in order to provide a feasible assembly. Because the size of each cluster is small, and because the repeats are not often present in G, such cyclic graphs are not common. Thus, in the majority of the cases, our assembler is guaranteed to produce an optimal assembly for a given cluster.

4.1.3.2. Breakpoint and Content Detection. The cluster assembly provides the sequence content. The insertion breakpoint can be inferred using the provided assembled contigs and the leftmost and the rightmost mapping locations kept for each cluster. Thus, to characterize the exact insertion breakpoint, we extract a piece from the reference sequence which encapsulates the leftmost and rightmost mapping locations of each cluster. We align the assembled contigs to the reference in the vicinity of each cluster using a modified variant of Smith-Waterman [93] algorithm. In the variant Smith-Waterman algorithm, we fix the left and right anchors of the assembled contig as in global alignment, yet we expect a local alignment on the reference sequence, i.e. the assembled contig is fully aligned to a substring of the genomic sequence (*global to local alignment*). When we are backtracking, we expect to see two flanks (left and right) and a big gap on the genomic side. In order to modify the Smith-Waterman aligner to be suitable for detecting long insertions we changed the alignment scores. We tried several scoring values on the simulation data and finally we set the match, mismatch, gap opening, and gap extension parameters of the aligner to 20, -1000, -1000 and -1, respectively. We only consider those candidate insertions that align to the reference by at least 6bp at both sides. In addition, either or both of the left and right flanks should align to the reference by at least 17bp. We finally cut out the left and right flanks of the contig agreeing with the reference and return the sequence between these two flanking sequences as the novel sequence insertion and the end of the left-mapping

flank as the exact breakpoint location (Figures 4.5 and 4.3D).

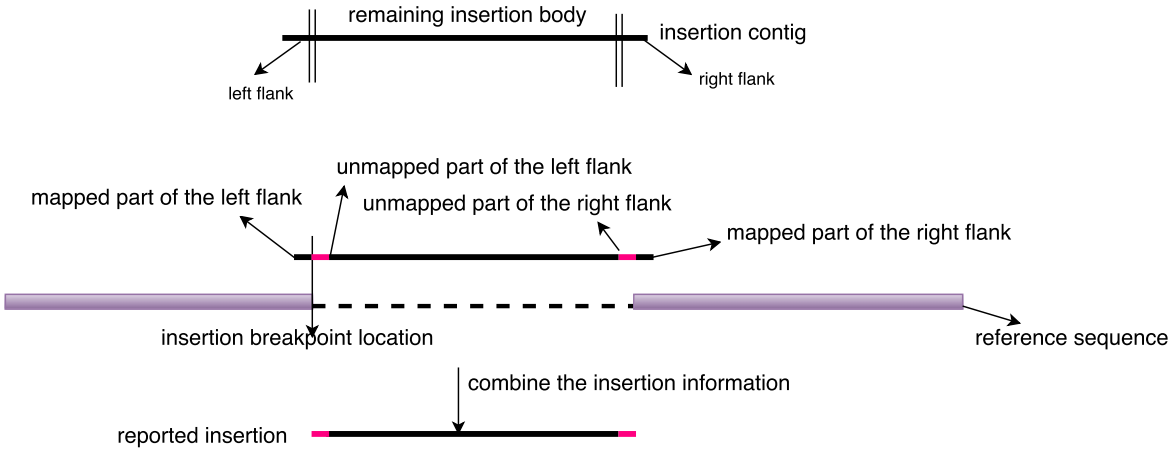

Figure 4.5. Exact breakpoint detection.

4.1.4. Post-processing and Genotyping

To refine our candidate list and eliminate false positives, for a dataset with fragment size L, we construct a temporary reference segment by concatenating three sequences:

(i) L bp upstream of the breakpoint from the reference;
(ii) the obtained insertion sequence from the previous step; and
(iii) L bp downstream of the breakpoint from the reference.

We then map all OEAs and orphan reads to this temporary reference. If there is a concordant mapping in which only one mate overlaps the insertion and the other mate is in the flanking region for both breakpoint locations of the insertion, this breakpoint passes our filter and we report the insertion. With this method, we guarantee that both breakpoints are covered by supporting reads, which are signatures of an insertion. If one of the flanking regions consists of Ns only, the other flanking region should meet the criteria. A false positive case will miss these reads and will be eliminated. Figure 4.6 represents a valid insertion and its reads are concordantly mapping on each side as described.

Figure 4.6. An example of a valid breakpoint passing the post-processing stage with concordantly mapping paired-end reads.

There might be still some reads which map to multiple novel insertions. We assign each such read to the insertion with the highest support via set-cover algorithm, where the set of reads represents the universe, and where clusters represent the sets. By selecting the minimal number of sets which describe all of the available reads, we eliminate low-support insertions and ensure that each read belongs to only one insertion event. Because the set cover problem is an NP-hard problem, we use a fast greedy strategy to calculate the minimal set of events that covers all reads [94].

Finally, we perform a genotype inference from the reported sequences as follows. We first construct the following two temporary sequences:

- I: concatenation of (i), (ii) and (iii) as the temporary reference that contains the novel sequence as described above; and
- RE: concatenation of (i) and (iii) as the temporary reference that does not contain the insertion.

We then align *all* reads to these two temporary reference sequences.

Let re be the number of reads that align across the breakpoint location in RE and i_l, i_r be the number of reads that align across, respectively, the left and right breakpoint locations in I. Figure 4.7 shows RE, I, and the calculation of i and re. We then predict the genotype using the Equation 4.2 below. In Figure 4.7, we show an example for calculating re, i, and x based on the Figure: RE**:** L bp upstream of the breakpoint on the reference + L bp downstream of the breakpoint on the reference; I**:** L bp upstream of the breakpoint on the reference + the insertion sequence + L bp downstream of the breakpoint on the reference; $re = 2$ (the # of mappings passing

through the breakpoint on RE); $i_l = 9$ (the # of mappings passing through the left breakpoint on I); $i_{re} = 7$ (the # of mappings passing through the right breakpoint on I); $i = (i_l + i_r)/2 = 8$; $\mathbf{x} = (i\text{-}re)/(i+re) = 0.6$. We tested various values for γ and we found $\gamma = 0.3$ yielded the best genotyping accuracy in simulated data. We report the final set of calls in standard VCF format [95].

$$i = \frac{i_l + i_r}{2}, \quad x = \frac{i - re}{i + re} \qquad \text{Genotype} = \begin{cases} \text{No Insertion} & \text{if } x \leq -\gamma \\ \text{Homozygous} & \text{if } x \geq \gamma \\ \text{Heterozygous} & \text{otherwise} \end{cases} \tag{4.2}$$

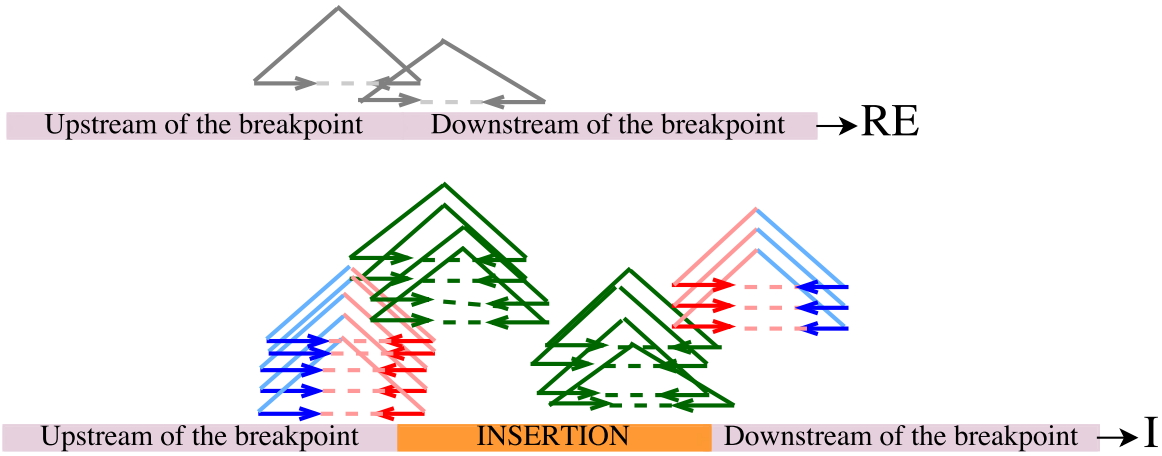

Figure 4.7. Genotyping novel sequence insertions with Pamir.

4.1.5. Discovery with Pooled Data

For multi-sample analysis (i.e. "pooled calling") Pamir maps all raw reads to the reference individually in best mapping mode, it extracts OEAs and orphans individually, and then it combines OEAs and orphans from all genomes, applies the same strategy until genotyping step and provides the list of insertions which exist in any or all of the genomes. After the initial discovery step, it genotypes each insertion for each sample by using individual reads. The steps for insertion discovery with multiple samples are depicted in Figure 4.8.

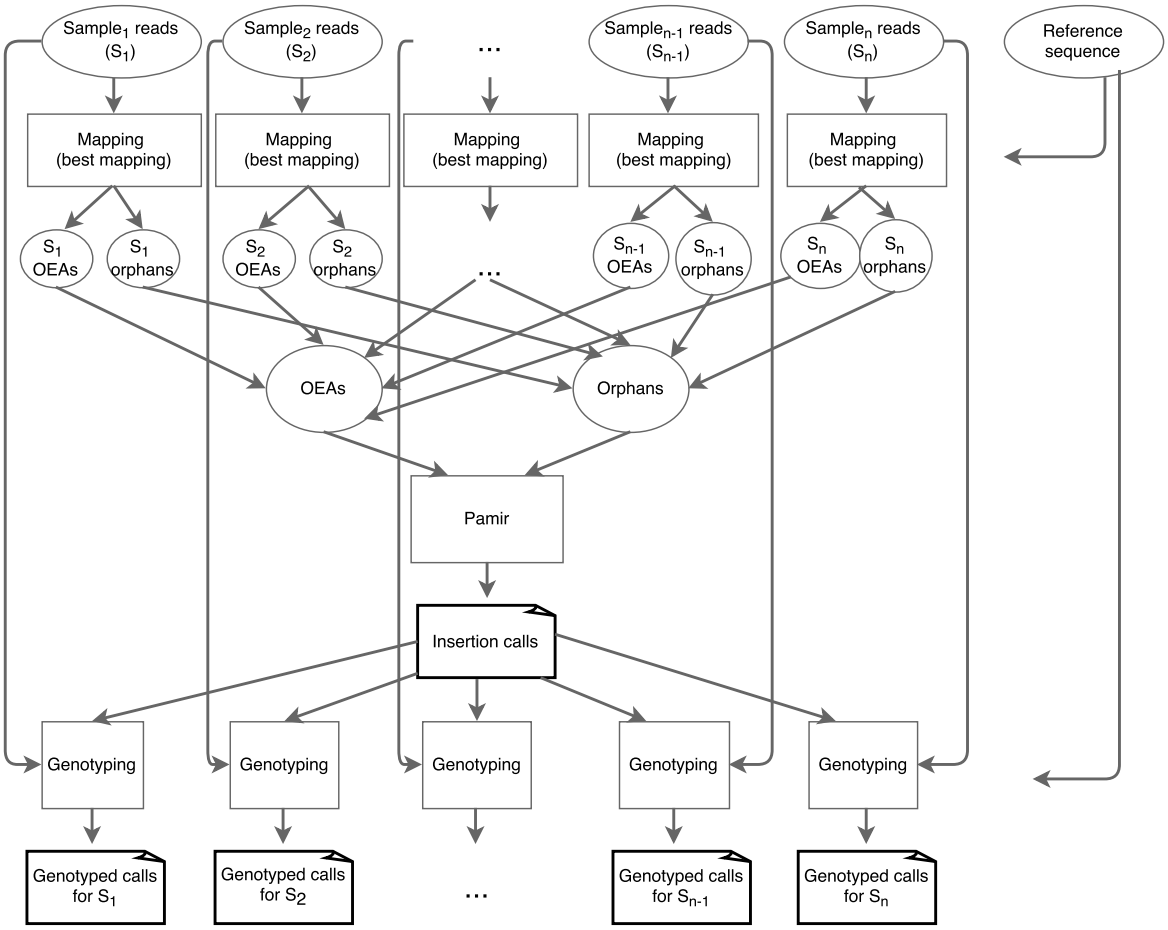

Figure 4.8. Discovery with pooled data.

4.2. Availability

Pamir is available at [96].

4.3. Results

We performed four sets of experiments to evaluate our method: two experiments with simulated data, and two experiments using real data. In simulation experiments, we inserted 350 new sequences into chromosome 21 of the GRCh37 reference in 7 different size ranges:

(i) 10–100bp,
(ii) 100–200bp,
(iii) 200–500bp,
(iv) 500–1,000bp,
(v) 1,000–2,000bp,
(vi) 2,000–5,000bp, and
(vii) 5,000–10,000bp.

We used randomly selected segments from the *Methylobacterium* reference genome for this purpose, which are guaranteed to be missing in the human genome reference. Next we generated several WGS read datasets using the ART read simulator [97] to test Pamir under different conditions:

(i) error-free reads generated as
    (a) 100bp Illumina HiSeq 2000,
    (b) 100bp Illumina HiSeq 2500,
    (c) 150bp Illumina HiSeq 2500;
(ii) noisy reads (i.e. introduced small variants as SNPs and Indels and sequencing errors. Default values of ART simulator are used.) generated as
    (a) 100bp Illumina HiSeq 2000,
    (b) 100bp Illumina HiSeq 2500, and

(c) 150bp Illumina HiSeq 2500.

All simulated data were created at 30x sequence coverage.

We also evaluated the efficacy of Pamir on low-coverage multi-sample data. For this purpose we simulated WGS data from five "samples" at 10x sequence coverage (noisy 100bp Illumina HiSeq 2500), with different novel sequence insertions. The "genome" of the first sample includes all 350 insertions described above, and we inserted 280 insertions to the other four samples. In all single-sample simulation experiments we compared Pamir with MindTheGap and PopIns. In multi-sample datasets, we compared Pamir with PopIns, which is the only tool capable of finding insertions in multi-sample data.

We tested Pamir on real datasets in two experiments. First, we applied Pamir on a high coverage WGS dataset generated from a single haploid sample (CHM1) [4] and compared our results with novel insertions found in the same genome with the SMRT-SV algorithm that uses long read, i.e. Pacific Biosciences, sequencing technology. Finally, we evaluated Pamir's performance in multi-sample insertion discovery and genotyping using 10 low-coverage WGS datasets generated as part of the 1000 Genomes Project [78].

4.3.1. Simulations

4.3.1.1. High coverage single sample. We summarize the results for the perfect simulation experiment with Illumina HiSeq2000 with 100bp reads as shown in Table 4.1. Recall rates are mostly lower in the longer insertion size regions and the recall rate of the last segment(5Kbp–10Kbp) is the lowest. Recall and precision rates are defined as follows: Precision is $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$, where TP is number of True Positives and FP is number of False Positives. Recall is $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, where TP is number of True Positives and FN is number of False Negatives. Comparison with PopIns, MindTheGap and BASIL-ANISE is given in Table 4.2. Briefly, Pamir outperforms MindTheGap, PopIns and BASIL-ANISE in all simulation experiments in terms of recall. In terms of precision

Table 4.1. Precision and recall rates of perfect Illumina HiSeq2000 simulation data distributed according to the insertion sizes.

Insertion Length	**# of Ins.**	**TP**	**FN**	**FP**	**Recall**	**Precision**
10-100bp	50	50	0	0	1.00	1.00
100-200bp	50	50	0	0	1.00	1.00
200-500bp	50	50	0	0	1.00	1.00
500-1Kbp	50	49	1	0	0.98	1.00
1Kbp-2Kbp	50	48	2	0	0.96	1.00
2Kbp-5Kbp	50	46	4	0	0.92	1.00
5Kbp-10Kbp	50	40	10	0	0.80	1.00
Total	350	333	17	0	0.95	1.00

Pamir outperforms all tools or has equal precision. The precision rates of Pamir and MindTheGap are the same for the first four datasets, and for the rest Pamir outperforms all other tools. Here we consider a predicted insertion to be correct only if the breakpoint matches that of the simulated insertion. In fact, if we also require the lengths of the predicted insertions to be the same with the simulation, Pamir has the best precision and recall among the tools we tested.

4.3.1.2. Low coverage multiple samples. Next, we tested the prediction performance of Pamir when multiple genomes with low coverage data are available. In this experiment we compared Pamir only with PopIns, as it is the only other multi-sample novel sequence insertion discovery tool. To evaluate the importance of multiple samples, we tested the same five genomes simulated at 10x sequence coverage both separately and collectively (Table 4.3). In the table, we show precision and recall rates of both individual and pooled calls of five low coverage samples. The paired-end reads (100bp) are generated using Illumina HiSeq2500 error model. We have simulated 350 insertions in this dataset: S_1 have all insertions, and genomes of the other four individuals contains 280 events. The column *All* shows performances of Pamir and PopIns based on pooling simulation reads, and each column S_i represents single sample detection results for i-th individual. We found that Pamir's precision was substantially higher than that of

Table 4.2. Precision and recall of Pamir, PopIns [1] and MindTheGap [2] and BASIL ANISE [3] on simulated 30x datasets generated for different sequencing platforms with varying read lengths. Best results are marked with bold typeface.

			Pamir	**PopIns**	**MindThe-Gap**	**BASIL-ANISE**
Error-free	HiSeq2500-100bp	Precision	**1.000**	0.973	**1.000**	0.989
		Recall	**0.954**	0.814	0.900	0.757
	HiSeq2500-150bp	Precision	**1.000**	0.958	**1.000**	0.989
		Recall	**0.960**	0.726	0.900	0.763
	HiSeq2000-100bp	Precision	**1.000**	0.972	**1.000**	0.989
		Recall	**0.951**	0.823	0.900	0.763
Noisy	HiSeq2500-100bp	Precision	**1.000**	0.969	**1.000**	0.989
		Recall	**0.926**	0.800	0.900	0.757
	HiSeq2500-150bp	Precision	**1.000**	0.968	0.965	0.989
		Recall	**0.943**	0.789	0.897	0.754
	HiSeq2000-100bp	Precision	**1.000**	0.938	0.905	0.974
		Recall	**0.826**	0.709	0.811	0.743

PopIns when each sample is processed separately, and use of multiple genomes resulted in higher recall rates for both tools.

We also predicted genotypes of all five samples using Pamir (Table 4.4). In the figure we show evaluation of genotyping results for the same five samples as in Table 4.3, based on pooling simulated reads. The paired-end reads (100bp) are generated using Illumina HiSeq2500 error model. We have simulated 350 insertions in this dataset: S_1 have all insertions, and genomes of the other four individuals contains 280 events. *Correct (INS)* lists the number of insertions that are correctly genotyped. *Correct (REF)* shows the number of detections discarded after genotyping, which are not actual insertions in an individual but falsely predicted based on pooling reads. *Incorrect zygosity* provides the number of insertions incorrectly genotyped as heterozygous; only 5 calls were identified as heterozygous in S_1, S_3 and $S4$ although they were homozygously inserted. All insertions map to common repeats. The *No call (INS)* row shows the

number of insertions missed in the pooled run for each sample, i.e. false negatives. *No call (REF)* provides the number of insertions missed in the pooled run but the insertion was not inserted into this sample. Here we first characterized insertions using all five samples simultaneously as described above, and then calculated genotypes for each predicted insertion in all samples separately. We observed no incorrect heterozygous vs. homozygous genotyping results for any insertions, except 5 calls in 3 samples are identified as heterozygous although they were homozygously inserted. All 5 insertions map to common repeats, i.e. LINE elements.

Table 4.3. Precision and recall rates of 5 simulated samples (noisy HiSeq2500 100bp 10x). Best results are marked with bold typeface.

Samples	**All**		S_1		S_2		S_3		S_4		S_5	
Experiment	**Pooled**		**Individual**		**Individual**		**Individual**		**Individual**		**Individual**	
# of Insertions	**350**		**350**		**280**		**280**		**280**		**280**	
Tools	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**
Precision	**1**	0.977	**1**	0.575	**1**	0.591	**1**	0.575	**1**	0.574	**1**	0.603
Recall	**0.911**	0.811	**0.726**	0.657	**0.711**	0.675	**0.704**	0.657	**0.714**	0.646	**0.714**	0.668

Table 4.4. Evaluation of predicted genotypes using 5 simulated genomes. Best results are marked with bold typeface.

Sample	S_1		S_2		S_3		S_4		S_5	
# of Insertions	**350**		**280**		**280**		**280**		**280**	
# of Ins. not in the sample	**0**		**70**		**70**		**70**		**70**	
Tools	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**	**Pamir**	**PopIns**
Correct (INS)	**317**	284	**253**	210	**252**	214	**253**	225	**259**	227
Correct (REF)	-	-	66	54	66	56	64	59	60	57
Incorrect zygosity	2	0	0	0	1	0	2	0	0	0
No call (INS)	31	66	27	50	27	52	25	55	21	53
No call (REF)	-	-	4	16	4	14	6	11	10	13

4.3.2. Real Data

4.3.2.1. High coverage sequencing of CHM1. Our tests using real data also included two types of datasets: (i) high coverage single sample WGS, and (ii) low coverage multiple sample WGS. First, we evaluated Pamir using WGS data at 40x coverage generated from a haploid cell line with the Illumina technology (CHM1, SRA ID: SRX652547) [4]. We have identified a total of 22,676 insertions that corresponds to 593,5 Kb in total, of which, 2,444 were >50bp (348 Kb total) (Table 4.5). Chaisson

et al. (2015) also generated *de novo* assembly of the same genome using a long read sequencing technology (Pacific Biosciences) from the same cell line, and predicted the insertions with the SMRT-SV algorithm using this dataset [4]. Here we used an updated call set (> 50bp) mapped to human GRCh38 for comparisons. Here, we used an updated call set (> 50bp) mapped to human GRCh38 [98] for comparisons.

Pamir showed low recall rates when compared to the long read-based SMRT-SV results [4]. We could identify only 488 of the 12,998 insertions detected by SMRT-SV when we consider only nearby matches (less than 10bp distance) in breakpoint predictions. One of the reasons for such discrepancy is the fact that more than half of PacBio-predicted insertions are located within various repeat regions (Table 4.6), and short-length Illumina reads are not sufficient to properly assemble such regions. The same effect was also observed in the original publication [4], where only a handful of insertions were also identified in another assembly of the same genome that was constructed with a reference-guided methodology using both Illumina WGS and bacterial artificial chromosome datasets [99]. We also compared the results of PopIns with SMRT-SV results in Table 4.7 and MindTheGap results with SMRT-SV results in Table 4.8. The common breakpoint locations between PopIns and SMRT-SV were low too same as PopIns SMRT and PopIns MindTheGap. Additionally, we found that 14,121 out of our 22,646 predicted insertions were reported in dbSNP version 147, considering 10bp breakpoint resolution.

To test whether the insertions we predicted in CHM1 were also previously discovered in other studies, we mapped the longer insertions (>50 bp) to the latest version of the reference (GRCh38) using BLAST [90]. Note that our predictions were based on the GRCh37 version. In this experiment we required only highly identical ($\geq$ 98%) hits that covered at least 98% of the predicted insertion. We repeated the same remapping experiment to both the long read-based assembly [4] and the alternative reference-guided assembly of the same genome [99] (Table 4.9). In the table, we provide a hierarchical non-redundant breakdown of comparison of insertions we predicted in the CHM1 genome with Pamir. We compare our predictions in the following order: the GRCh38 assembly, then remaining to the reference-guided CHM1_1.1 assembly, the

Pacific Biosciences (PacBio) assembly, SMRT-SV call set, long insert clones and those that are in repeat regions. We also mapped the same sequences to the `nt/nr` database to detect whether the sequences were also contained within other WGS studies, in particular, fosmid end-sequence data [100]. In summary, out of 2,444 (> 50 bp) insertions we predicted, 1,418 are not found in any database, of which 1,189 were mapped to common repeats. We applied the same experiment on PopIns and MindTheGap calls too (Tables 4.10 and 4.11). In these tables, same strategy as in Table 4.9 is used. We provide a hierarchical non-redundant breakdown of comparison of insertions we predicted in the CHM1 genome with PopIns. We compare PopIns predictions in the following order: the GRCh38 assembly, then remaining to the reference-guided CHM1_1.1 assembly, the Pacific Biosciences (PacBio) assembly, SMRT-SV call set, long insert clones and those that are in repeat regions. According to the results of this experiment, more insertions predicted by PopIns were seen in the latest version of GRCh38, reference-guided assembly of CHM1 [99] and long read-based assembly [4]. When we analyzed the insertions predicted by PopIns in detail, we saw that many of them were reported at different chromosomes, at different locations of chromosomes or they were partial insertions compared to [98] results. Therefore, we needed to do a more strict experiment to compare these insertion calls. We mapped the longer insertions (>50 bp) with spanning regions around the breakpoint on the reference (GRCh37) to the latest version of the reference (GRCh38) using BLAST [90]. In summary, according to the more strict experiment, out of 2,444 (> 50 bp) insertions we predicted with Pamir, 1,446 are not found in any database, of which 1,212 mapped to common repeats (Table 4.12). In the table, we provide a hierarchical non-redundant breakdown of comparison of insertions we predicted in the CHM1 genome with Pamir with spanning regions. We compare our predictions in the following order: the GRCh38 assembly, then remaining to the reference-guided CHM1_1.1 assembly, the Pacific Biosciences (PacBio) assembly, SMRT-SV call set, long insert clones and those that are in repeat regions. We performed the same experiment using PopIns (Table 4.13). Same as Table4.12, in this table, we provide a hierarchical non-redundant breakdown of comparison of insertions we predicted in the CHM1 genome with PopIns with spanning regions. We compare PopIns predictions in the following order: the GRCh38 assembly, then remaining to

the reference-guided CHM1_1.1 assembly, the Pacific Biosciences (PacBio) assembly, SMRT-SV call set, long insert clones and those that are in repeat regions. 1,014 out of 3,399 PopIns calls are not found in any database, of which 388 mapped to common repeats. 56% of PopIns calls map to long insert clones, but only a handful were included in the latest version of the human genome reference, and assemblies of the same DNA resource.

Finally, we compared our results with PopIns [1] and MindTheGap [2] (Table 4.14 and Table 4.15). In Table4.15, we consider 10bp breakpoint resolution while comparing Pamir, PopIns and MindTheGap breakpoint locations. We don't compare the lengths of the insertions, which may differ one tool's callset to another. Insertion length ranges in Total column show that Pamir reports them in that length range. "same range" shows the number of insertions reported by the corresponding tool in the same length range. "diff range" shows the number of insertions reported by the corresponding tool in a different length range. For example, in the first row, Pamir shares 6 breakpoint locations with PopIns and Pamir reports all of them in $1 - 50$ bp length range and PopIns reports all of them in a different length range. The number of shared insertions between Pamir and PopIns is very low. Similarly, MindTheGap and PopIns share few number (108) of insertions too (not shown in the table). Finally, 40 of Pamir - PopIns intersection insertions are seen in MindTheGap - PopIns intersection.

4.3.2.2. High coverage sequencing of NA12878. As a second high coverage single sample WGS we ran Pamir on the sample NA12878 from 1000genome data sequenced at $\sim$ 200x coverage. We have identified a total of 78,996 insertions that corresponds to 4,54Mb in total. 21,379 of them were >50bp (3,75Mb total). The results are shown in Table 4.16. In order to investigate whether the called insertions do also exist in the existing databases we compared the insertion breakpoint locations (considering 10bp breakpoint resolution) with Genome in a Bottle (GIAB) [101] results, 1000 Genomes Project [78] calls and dbSNP147 [102] in a hierarchical non-redundant breakdown order (Table 4.17). We also compared the results of NA12878 with Pamir and with PopIns in Table 4.18. In this table, we consider 10bp breakpoint resolution while comparing

Pamir and PopIns breakpoint locations. We don't compare the lengths of the insertions, which may differ one tool's callset to another. Insertion length ranges in Total column show that Pamir reports them in that length range. "same range" shows the number of insertions reported by the corresponding tool in the same length range. "diff range" shows the number of insertions reported by the corresponding tool in a different length range. For example, in the first row, Pamir shares 7 breakpoint locations with PopIns and Pamir reports all of them in $1 - 50$ bp length range and PopIns reports all of them in a different length range. In order to compare with MindTheGap, we also tried to run MindTheGap on NA12878 but we were not able to obtain a resulting callset.

4.3.2.3. Low coverage genomes from the 1000 Genomes Project. Finally, we tested Pamir using low coverage WGS datasets generated from 10 samples as part of the 1000 Genomes Project [78] (Table 4.19). We found 39,554 insertions when we pooled all 10 genomes, 13,255 of them were reported in 1000 Genomes Project, and another group of 11,019 insertions was seen in dbSNP version 147, considering 10bp breakpoint resolution. We then genotyped for each sample (Table 4.20). To test whether the insertions we predicted in these 10 samples were also previously discovered in other studies, we mapped the longer insertions (>50 bp) to the latest version of the reference (GRCh38) using BLAST [90]. We also mapped the same sequences to the `nt/nr` database (Table 4.21). In this table, we provide a hierarchical non-redundant breakdown of comparison of insertions we predicted in the 10 1000 genomes. We compare our predictions in the following order: the GRCh38 assembly, then remaining to the long insert clones and those that are in repeat regions.

4.3.3. Running Times

Finally, we evaluated the running time of all the benchmarked software. We ran Pamir, PopIns, MindTheGap and BASIL-ANISE on a 800Mhz AMD machine with 256Gb memory with 1 thread on a high coverage simulation dataset (2x100bp error-free reads sampled from human chromosome 21 based on Illumina HiSeq2500 model at 30X

Table 4.5. Summary of insertions predicted in CHM1 with Pamir.

	All	**$\leq$ 50bp**	**$>$ 50bp**
Number of insertions	22,676	20,232	2,444
Minimum length	5	5	51
Maximum length	4,135	50	4,135
Average length	26.20	12.12	142.51
Heterozygous	2,281	1,765	516
Homozygous	20,395	18,467	1,928

Table 4.6. Comparison of insertions in CHM1 predicted using Illumina reads with Pamir and PacBio reads with SMRT-SV [4].

Size range	**Illumina + Pamir**	**PacBio + SMRT-SV [4]**	**Shared**	**SMRT-SV insertions in repeat regions**
1 - 50 bp	20,232	187 (all 50bp)	27	112
50 - 100 bp	1,273	4,384	205	2,358
100 - 200 bp	815	2,959	125	1,604
200 - 500 bp	291	3,123	97	1,707
>500 bp	65	2,345	34	1,411
All	22,676	12,998	488	7,192

coverage) until genotyping phase. Running times are given in Table 4.22. Pamir takes $\sim$ 3.6 times less time than BASIL-ANISE and $\sim$ 4.3 times less time than MindTheGap where PopIns takes $\sim$ 5.7 times less time than BASIL-ANISE and $\sim$ 6.8 times less time than MindTheGap. Although PopIns is faster than Pamir, in many of the cases it does not provide the full inserted sequences. Pamir's structure is embarrassingly parallel where the user can declare the number of threads according to the number of cores of the machine. We also tested the timing performance of Pamir on whole genome real data. It takes Pamir $\sim$ 16 hours to complete the insertion discovery on the whole genome of CHM1 real data (40X) with 72 threads.

Table 4.7. Comparison of insertions in CHM1 predicted using Illumina reads with PopIns and PacBio reads with SMRT-SV [4].

Size range	**Illumina + PopIns**	**PacBio + SMRT-SV [4]**	**Shared**	**SMRT-SV insertions in repeat regions**
1 - 50 bp	21	187 (all 50bp)	0	112
50 - 100 bp	246	4,384	17	2,358
100 - 200 bp	793	2,959	116	1,604
200 - 500 bp	1,074	3,123	138	1,707
>500 bp	1,286	2,345	203	1,411
All	3,420	12,998	474	7,192

Table 4.8. Comparison of insertions in CHM1 predicted using Illumina reads with MindTheGap and PacBio reads with SMRT-SV [4].

Size range	**Illumina + MindTheGap**	**PacBio + SMRT-SV [4]**	**Shared**	**SMRT-SV insertions in repeat regions**
1 - 50 bp	23,955	187 (all 50bp)	34	112
50 - 100 bp	533	4,384	132	2,358
100 - 200 bp	555	2,959	90	1,604
200 - 500 bp	672	3,123	85	1,707
>500 bp	268	2,345	61	1,411
All	25,983	12,998	402	7,192

4.4. Discussion

The last few years since the introduction of HTS platforms witnessed the development of many algorithms that aim to characterize genomic structural variation. The first such algorithms focused mainly on the discovery of deletions, and other forms of complex SV, especially inversions and translocations were largely neglected due to the sequence complexity around their breakpoints and the ambiguity in mapping to these

Table 4.9. Analysis of predicted CHM1 insertions with Pamir with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	2,083	295	66	2,444
In GRCh38	134	1	1	136
In CHM1_1.1 [99]	353	48	0	401
in CHM1 PacBio [4]	138	11	21	170
In SMRT-SV [4]	94	53	13	160
In long insert clones* [100]	118	18	1	137
In repeat regions	1,039	127	23	1,189
Remainder	196	28	5	229

*Long insert clones include both fosmid clones and bacterial artificial chromosomes (BAC).

Table 4.10. Analysis of predicted CHM1 insertions with PopIns with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	1,039	1,074	1,286	3,399
In GRCh38	367	461	775	1,603
In CHM1_1.1 [99]	116	78	7	201
in CHM1 PacBio [4]	439	470	482	1,391
In SMRT-SV [4]	11	6	4	21
In long insert clones* [100]	12	23	5	40
In repeat regions	62	24	8	94
Remainder	32	7	5	49

*Long insert clones include both fosmid clones and bacterial artificial chromosomes (BAC).

regions.

Although novel sequence insertions can be considered "simpler" than most other SV classes, their accurate characterization is still lacking due to the need for constructing either global or local *de novo* assembly. However, they may fail to generate long and accurate contigs due to the repeats that may occur around or within novel sequence

Table 4.11. Analysis of predicted CHM1 insertions with MindTheGap with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	1,088	672	268	2,028
In GRCh38	173	42	35	250
In CHM1_1.1 [99]	177	38	4	219
in CHM1 PacBio [4]	61	7	23	91
In SMRT-SV [4]	52	53	35	140
In long insert clones* [100]	165	200	72	437
In repeat regions	313	169	68	550
Remainder	147	163	31	341

*Long insert clones include both fosmid clones and bacterial artificial chromosomes (BAC).

Table 4.12. (Strict version with 200bp spanning regions on the reference) Analysis of predicted CHM1 insertions with Pamir with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	2,088	291	65	2,444
In GRCh38	17	1	1	19
In CHM1_1.1 [99]	251	54	2	307
in CHM1 PacBio [4]	213	13	23	249
In SMRT-SV [98]	73	47	11	131
In long insert clones* [100]	212	21	1	234
In repeat regions	1,065	126	21	1212
Remainder	257	29	6	292

*Long insert clones include both fosmid clones and bacterial artificial chromosomes (BAC).

insertions.

In this stduy, we presented Pamir, a new algorithm to discover and genotype novel sequence insertions in one or multiple human genomes. Pamir uses several read signatures (one-end-anchored, read pairs, split reads, and assembly) to characterize insertions that span a wide size range. We demonstrated its performance on both

Table 4.13. (Strict version with 200bp spanning regions on the reference) Analysis of predicted CHM1 insertions with PopIns with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	1,038	1,075	1,286	3,399
In GRCh38	0	1	1	2
In CHM1_1.1 [99]	15	8	1	24
in CHM1 PacBio [4]	5	2	12	19
In SMRT-SV [98]	118	132	193	443
In long insert clones* [100]	565	627	705	1,897
In repeat regions	221	191	214	626
Remainder	114	114	160	388

*Long insert clones include both fosmid clones and bacterial artificial chromosomes (BAC).

Table 4.14. Insertions in CHM1 predicted with Pamir, PopIns [1] and MindTheGap [2].

Size range	**Pamir**	**PopIns [1]**	**MindTheGap [2]**
1 - 50 bp	20,232	21	23,955
50 - 100 bp	1,273	246	533
100 - 200 bp	815	793	555
200 - 500 bp	291	1,074	672
>500 bp	65	1,286	268
All	22,676	3,420	25,983

We consider 10bp breakpoint resolution.

simulated and real datasets and showed that it outperforms the existing tools designed for the same purpose. We believe that further development and extensive testing of the Pamir algorithm will help make the novel insertion discovery a routine analysis for whole genome sequencing studies.

Table 4.15. Comparison of breakpoint locations in CHM1 predicted with Pamir and with PopIns [1] and MindTheGap [2].

Size range	**Pamir**	**Shared w. PopIns [1]**			**Shared w. MindTheGap [2]**		
		Total	**Same range***	**Diff. range**	**Total**	**Same range**	**Diff. range**
1 - 50 bp	20,232	6	0	6	5,137	5,114	23
50 - 100 bp	1,273	8	2	6	106	82	24
100 - 200 bp	815	31	26	5	67	49	18
200 - 500 bp	291	33	23	10	26	17	9
>500 bp	65	22	20	2	15	12	3
All	22,676	100	71	29	5,351	5,274	77

*We consider 10bp breakpoint resolution while comparing Pamir, PopIns and MindTheGap breakpoint locations.

Table 4.16. Summary of insertions predicted in NA12878.

	All	**$\leq$ 50bp**	**$>$ 50bp**
Number of insertions	78,996	57,617	21,379
Minimum length	5	5	51
Maximum length	6,796	50	6,796
Average length	57.49	13.43	176.20
Heterozygous	25,366	19,528	5,838
Homozygous	53,630	38,089	15,541

Table 4.17. Comparison of NA12878 insertion calls with the existing databases.

	All	**$\leq$ 50bp**	**$>$ 50bp**
Number of insertions	78,996	57,617	21,379
Genome in a Bottle (GIAB) [101]*	12,354	11,420	934
In 1000 Genomes Project [78]	7,834	5,361	2,473
In dbSNP version 147*	19,769	15,769	4000

* We intersected with 1000 Genomes after removing those insertions found in GIAB and intersected with dbSNP database after removing those insertions that are found in 1000 Genomes.

Table 4.18. Comparison of insertions in NA12878 predicted with Pamir and with PopIns [1].

Size range	**Pamir**	**PopIns [1]**	**Shared**		
			Total	**Same range***	**Diff. range**
1 - 50 bp	57,617	2	7	1	6
50 - 100 bp	7,083	127	33	12	21
100 - 200 bp	9,055	992	71	52	19
200 - 500 bp	4,453	1,065	88	61	27
>500 bp	788	818	59	27	32
All	78,996	3,004	258	153	105

*We consider 10bp breakpoint resolution while comparing Pamir and PopIns breakpoint locations.

Table 4.19. Summary of novel sequences found in 10 low coverage WGS datasets from the 1000 Genomes Project.

	Total	**> 50bp**
Number of insertions	49,473	6,846
Minimum length	5	51
Maximum length	1,928	1,928
Average length	28.872	128.085
In 1000 Genomes Project [78]	14,837	425
In dbSNP version 147*	14,409	2,027

* We intersected with dbSNP after removing those insertions that are found in the 1000 Genomes Project.

Table 4.20. Genotyping results for the novel sequences found in the 1000 Genomes Project datasets.

	06985*	**07357**	**10851**	**11840**	**11918**	**11933**	**12004**	**12044**	**12234**	**12286**
Homozygous	22,971	22,582	23,274	20,973	22,610	21,049	19,024	18,753	20,841	19,027
Heterozygous	10,246	10,158	9,465	12,745	9,994	11,092	12,650	13,002	10,804	12,622
Total insertion length (bp)	941,868	921,225	930,766	959,017	953,968	936,615	928,371	919,212	916,251	922,799

* All sample IDs start with "NA".

Table 4.21. Analysis of insertions found in low-coverage samples with respect to other datasets.

	50 - 200 bp	**200 - 500 bp**	**>500 bp**	**Total**
# of insertions	6,050	667	129	6,846
in GRCh38	377	7	1	385
in long insert clones [100]	868	84	31	983
in repeat regions	3,614	464	69	4,147
Remainder	1,191	112	28	1,331

Table 4.22. Running times of Pamir, PopIns, MindTheGap, and BASIL-ANISE on a 2x100bp simulation dataset based on HiSeq2500 model with 30X coverage.

Pamir	**PopIns**	**MindTheGap**	**BASIL-ANISE**
3min 9sec	1min 59sec	13min 25sec	11min 16sec

5. CONCLUSION

In this thesis, we presented our research on three different problems in the genomics field. The ability of high throughput sequencing technologies to generate tremendous amount of data for very low costs enabled many large scale sequencing projects possible. It also increased the research to analyze the huge amounts of data. Reproducibility and robustness of the data are the essential priorities in clinical applications. HTS data are assumed to be more robust and comprehensive and there are many studies testing the performance of HTS platforms, but the robustness has not been tested much. In our first study, we presented our investigation on the robustness of an HTS platform, Illumina HiSeq2000, in terms of being directly used in clinical applications. We whole genome sequenced the genomes of two individuals twice with the same model of HTS platform, Illumina HiSeq2000, placed at two different locations: BGI in Beijing, China and TÜBİTAK in Kocaeli, Turkey. We used the same tools and parameters to call the small structural variations (SNP-Indels) and expected to see the same results but it was not the case. We discovered that the reproducibility rate of SNP calls is $\sim$ 92%, which are obtained with the standard BWA mapping and GATK SNP calling pipeline. This also means that $\sim$280 thousand of 3 million SNPs are unique to only one of the trials. Pooled calling with GATK did not improve the reproducibility and accuracy of the results substantially. Multiple factors contribute to this effect such as the difference in GC% bias which leads to differences in the coverage over different regions of the genome, which then causes increasing number of false positives or false negatives in variation calls. Higher GC% content in repeat regions than unique regions, in addition to the mapping biases to repeats and duplications also explain the high SNP discrepancies in repeat regions. In addition since the machines are individually different, they might have slight differences in base calling errors. Most of the differences were in non-genic regions and common repeats which is less important to most o fthe studies but still one must be very careful when interpreting results from HTS pipelines especially for clinical diagnosis. The results should be validated with orthogonal studies.

As a second problem we worked on improving the *de novo* genome assemblies which is one of the two main algorithmic problems in this area and offered a new method. Genome assembly has not been improved as much as genome alignment since a reference genome is not used, and the genome itself is repetitive and duplicated and the HTS reads are either short or long but with higher error rate. Short reads cannot assemble the repetitive regions no matter how accurate they are, and long reads can assemble the repetitive regions but they are not accurate enough having high sequencing error rates. With our method, we showed that by exploiting both short and highly accurate and long but erroneous reads we can improve the resulting assembly accuracy and coverage. We used short Illumina reads, long 454 reads and medium sized Ion Torrent reads, assembled them separately and corrected the long read contigs by short read contigs which gave more accurate and continuous *de novo* assembly and our results showed that and using three data types all together further improves the results. Correction with two data types or three data types both give more accurate results than running the two other existing hybrid assemblers with the raw data. For further study the gaps between the updated contigs can be filled with the help of high quality paired-end information.

Finally we worked on a problem which did not get as much attention as other structural variations (SV) such as deletions in the genome: finding and genotyping novel sequence insertions in the whole genome sequencing data. One of the reasons that the problem did not get as much attention was the difficulties in the assembly itself. We tried to solve this problem with two different stages of assembly, first is a regular de-Bruijn graph based assembly of orphan reads and the second is a strand aware OLC assembly of the orphan contigs and OEA reads together which are both contributing to the novel insertion. As a result of our studies we presented our tool Pamir which identifies long novel sequence insertions in one or many human genomes with a new algorithm. It exploits the unmapped reads of the data (one-end anchors, split reads and orphan reads), and it is able to detect the exact breakpoint location of the insertion, the exact sequence of the insertion, and the genotype of the insertion (homozygous, heterozygous or no insertion). It can detect insertions in low coverage genomes by making a pooled run where it combines the data from all genomes. Our

simulated experiments show that it outperforms MindTheGap and PopIns on single genome and again outperforms PopIns when ran on multiple input data.

We are glad to mention that our studies are valuable contributions to the research in HTS technologies, the *de novo* assembly problem and discovering novel insertions in whole genome sequencing studies. Two last two studies contribute to have more accurate and complete assembly sequences. The last study also contributes to find the unknown genomic regions of the human reference sequence.

6. FUTURE WORK

In terms of investigating the robustness of the sequencing platforms, the method followed in this thesis was only one of many ways of assessing the robustness of the Illumina HiSeq2000 high throughput sequencing platform data with BWA and GATK pipeline. For example the make and model of the machines used in our experiment were same but they were individually different machines, which might cause slight differences in the sequencing output. The test can be re-applied by sequencing the same DNA with exactly the same machine to get rid of any differences caused by individually different machines. Also, here we focused on only SNP-indel calling which is a routine analysis in clinical applications. The robustness of the detecting large structural variations also needs to be investigated.

In addition, independent from the sequencing machines, a recent study [39] showed that BWA generates different mapping results even with the same but reshuffled set of reads, because of mapping biases through repeat regions. Although sequence alignment is simpler and more useful than before, decreasing the mapping biases against repeat regions in sequence alignment is another topic of interest for future studies. Same applies for variant calling, because it is shown that GATK also generates different callsets given the same mapping files [39].

There is potential future work for improving genome assemblies. Although, the results showed that correction method presented in this thesis works very well to improve genome assemblies, the need to develop new methods that exploit different data properties of different HTS technologies, such as short/long reads or high/low quality of reads, remains. For example, there were still gaps between the contigs even after the correction method. Regarding this, exploiting the paired end information of the short, high quality reads after the correction phase to extend the corrected contigs and to close the gaps between them would be a reasonable future work to improve the method.

Discovering novel sequence insertions on whole genome data has also interesting future work. Although our method, Pamir outperforms the existing tools it still lacks very long insertions ($>$ 10.000bp) due to breaks in the assembly caused by repetitive structures. Multiple shorter contigs instead of one continuous contig are obtained from the orphan assembly. Scaffolding with paired-end data information might be a good way of extending and connecting orphan contigs which will increase the recall rate on longer insertions. In addition, future work might also include improving the genotyping algorithm by replacing the simplistic genotyping formula used in Pamir with a more complicated genotype likelihood algorithm. Lastly, the multi-sample support is working well for now, but orphan assembly from many samples might be a restrictive task in terms of memory requirements. Potential future work can be to improve the orphan assembly by applying it on groups of limited number of samples and combining the resulting orphan contigs.

REFERENCES

1. Kehr, B., P. Melsted and B. V. Halldorsson, "PopIns: population-scale detection of novel sequence insertions", *Bioinformatics*, April 2015.

2. Rizk, G., A. Gouin, R. Chikhi and C. Lemaitre, "MindTheGap: integrated detection and assembly of short and long insertions.", *Bioinformatics*, Vol. 30, No. 24, pp. 3451–3457, December 2014.

3. Holtgrewe, M., L. Kuchenbecker and K. Reinert, "Methods for the detection and assembly of novel sequence in high-throughput sequencing data.", *Bioinformatics*, Vol. 31, No. 12, pp. 1904–1912, February 2015.

4. Chaisson, M. J. P., J. Huddleston, M. Y. Dennis, P. H. Sudmant, M. Malig, F. Hormozdiari, F. Antonacci, U. Surti, R. Sandstrom, M. Boitano, J. M. Landolin, J. A. Stamatoyannopoulos, M. W. Hunkapiller, J. Korlach and E. E. Eichler, "Resolving the complexity of the human genome using single-molecule sequencing.", *Nature*, Vol. 517, pp. 608–611, January 2015.

5. Maxam, A. M. and W. Gilbert, "A new method for sequencing DNA", *Proceedings of National Academy of Sciences of the USA*, Vol. 74, pp. 560–564, August 1977.

6. Sanger, F., S. Nicklen and A. Coulson, "DNA Sequencing with chain terminating inhibitors", *Proceedings of the National Academic of Sciences of the USA*, Vol. 74, pp. 5463–5467, December 1977.

7. Manning, M. and L. Hudgins, "Array-based technology and recommendations for utilization in medical genetics practice for detection of chromosomal abnormalities", *Genet Med*, Vol. 12, No. 11, pp. 742–745, 2010.

8. Miller, J. R., S. Koren and G. Sutton, "Assembly Algorithms for Next Generation Sequencing Data", *Genomics*, Vol. 95, No. 6, pp. 315–327, June 2010.

9. Lam, H. Y. K., M. J. Clark, R. Chen, R. Chen, G. Natsoulis, M. O'Huallachain, F. E. Dewey, L. Habegger, E. A. Ashley, M. B. Gerstein, A. J. Butte, H. P. Ji and M. Snyder, "Performance comparison of whole-genome sequencing platforms", *Nature Biotechnology*, Vol. 30, pp. 78–82, 2012.

10. Loman, N. J., R. V. Misra, T. J. Dallman, C. Constantinidou, S. E. Gharbia, J. Wain and M. J. Pallen, "Performance comparison of benchtop high-throughput sequencing platforms", *Nature Biotechnology*, Vol. 30, pp. 434–439, May 2012.

11. Zook, J. M., B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide and M. Salit, "Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls.", *Nat Biotechnol*, Vol. 32, No. 3, pp. 246–251, March 2014.

12. Venter, J. C. e. a., "The Sequence of Human Genome", *Science*, Vol. 291, No. 5507, pp. 1304–1351, February 2001.

13. Alkan, C., B. P. Coe and E. E. Eichler, "Genome structural variation discovery and genotyping.", *Nat Rev Genet*, Vol. 12, No. 5, pp. 363–376, May 2011.

14. Chaisson, M. J. P., R. K. Wilson and E. E. Eichler, "Genetic variation and the de novo assembly of human genomes.", *Nat Rev Genet*, Vol. 16, pp. 627–640, 2015.

15. Kidd, J. M., N. Sampas, F. Antonacci, T. Graves, R. Fulton, H. S. Hayden, C. Alkan, M. Malig, M. Ventura, G. Giannuzzi, J. Kallicki, P. Anderson, A. Tsalenko, N. A. Yamada, P. Tsang, R. Kaul, R. K. Wilson, L. Bruhn and E. E. Eichler, "Characterization of missing human genome sequences and copy-number polymorphic insertions.", *Nat Methods*, Vol. 7, No. 5, pp. 365–371, May 2010.

16. Chennagiri, N., E. J. White, A. Frieden, E. Lopez, D. S. Lieber, A. Nikiforov, T. Ross, R. Batorsky, S. Hansen, V. Lip, L. J. Luquette, E. Mauceli, D. Margulies, P. M. Milos, N. Napolitano, M. M. Nizzari, T. Yu and J. F. Thompson,

"Orthogonal NGS for High Throughput Clinical Diagnostics", *Nature Scientific Reports*, Vol. 6, April 2016.

17. Myers, E. W., G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams and J. C. Venter, "A whole-genome assembly of Drosophila.", *Science*, Vol. 287, No. 5461, pp. 2196–2204, March 2000.

18. Miller, J. R., A. L. Delcher, S. Koren, E. Venter, B. P. Walenz, A. Brownley, J. Johnson, K. Li, C. Mobarry and G. Sutton, "Aggressive assembly of pyrosequencing reads with mates", *Bioinformatics*, Vol. 24, No. 24, pp. 2818–2824, October 2008.

19. Zimin, A., G. Marçais, D. Puiu, M. Roberts, S. L. Salzberg and J. A. Yorke, "The MaSuRCA genome Assembler", *Bioinformatics*, Vol. 29, No. 21, pp. 2669–2677, August 2013.

20. G. Frey, K., J. H.-G. Enrique, C. L. Redden, T. V. Luu, S. L. Servetas, A. J. Mateczun, V. P. Mokashi and K. A. Bishop-Lilly, "Comparison of three next-generation sequencing platforms for metagenomic sequencing and identification of pathogens in blood", *BMC Genomics*, Vol. 15, No. 4, February 2014.

21. Quail, M. A., M. Smith, P. Coupland, T. D. Otto, S. R. Harris, T. R. Connor, A. Bertoni, H. P. Swerdlow and Y. Gu, "Comparison of mapping algorithms used in high-throughput sequencing: application to Ion Torrent data", *BMC Genomics*, Vol. 13, No. 1, p. 341, July 2012.

22. Caboche, S., C. Audebert, Y. Lemoine and D. Hot, "Comparison of mapping algorithms used in high-throughput sequencing: application to Ion Torrent data", *BMC Genomics*, Vol. 15, No. 1, p. 264, April 2014.

23. Alkan, C., P. Kavak, M. Somel, O. Gokcumen, S. Uğurlu, C. Saygı, E. Dal, K. Buğra-Bilge, T. Güngör, S. C. Sahinalp, N. Özören and C. Bekpen, "Whole genome sequencing of 16 Turkish genomes reveals functional private alleles and impact of genetic interactions with Europe, Asia and Africa.", *BMC Genomics*, Vol. 15, No. 963, November 2014.

24. DePristo, M. A., E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. del Angel, M. A. Rivas, M. Hanna, A. McKenna, T. J. Fennell, A. M. Kernytsky, A. Y. Sivachenko, K. Cibulskis, S. B. Gabriel, D. Altshuler and M. J. Daly, "A framework for variation discovery and genotyping using next-generation DNA sequencing data.", *Nature genetics*, Vol. 43, No. 5, pp. 491–498, May 2011.

25. Li, H. and R. Durbin, "Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform", *Bioinformatics*, Vol. 25, No. 14, pp. 1754–1760, Jul. 2009.

26. Li, H., B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin and . G. P. D. P. Group, "The Sequence Alignment/Map format and SAMtools", *Bioinformatics*, Vol. 25, No. 16, pp. 2078–2079, May 2009.

27. Quinlan, A. R. and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features.", *Bioinformatics*, Vol. 26, No. 6, pp. 841–842, March 2010.

28. Simon, A., "FastQC: A Quality Control tool for High Throughput Sequence Data", `http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/`, accessed at February 2017.

29. Alkan, C., J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, S. C. Sahinalp, R. A. Gibbs and E. E. Eichler, "Personalized copy number and segmental duplication maps using next-generation sequencing.", *Nature genetics*, Vol. 41, No. 10, pp. 1061–1067,

October 2009.

30. Hormozdiari, F., C. Alkan, E. E. Eichler and S. C. Sahinalp, "Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes", *Genome Research*, Vol. 19, No. 7, pp. 1270–1278, May 2009.

31. Wang, K., M. Li and H. Hakonarson, "ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data", *Nucleic Acids Research*, Vol. 38, No. 16, p. e164, September 2010.

32. Kavak, P., B. Yüksel, S. Aksu, M. O. Kulekci, T. Güngör, F. Hach, S. C. Şahinalp, T. H. G. Project, C. Alkan and M. S. Sağıroğlu, "Robustness of Massively Parallel Sequencing Platforms", *PLOS ONE*, Vol. 10, No. 9, pp. 1–11, September 2015.

33. "Robustness of Massively Parallel Sequencing Platforms Scripts directory", `https://github.com/pinarkavak/robust`, accessed at January 2017.

34. "Robustness of Massively Parallel Sequencing Platforms VCF files directory", `https://github.com/pinarkavak/robust`, accessed at January 2017.

35. "UCSC Genome Browser", `http://genome.ucsc.edu`, accessed at January 2017.

36. Consortium, T. . G. P., "An integrated map of genetic variation from 1,092 human genomes", *Nature*, Vol. 491, pp. 56–65, November 2012.

37. Kung, J. T. Y., D. Colognori and J. T. Lee, "Long Noncoding RNAs: Past, Present, and Future", *Genetics*, Vol. 193, No. 3, pp. 651–669, March 2013.

38. Biesecker, L. G., J. C. Mullikin, F. M. Facio, C. Turner, P. F. Cherukuri, R. W. Blakesley, G. G. Bouffard, P. S. Chines, P. Cruz, N. F. Hansen, J. K. Teer, B. Maskeri, A. C. Young, N. I. S. C. C. S. P. , T. A. Manolio, A. F. Wilson, T. Finkel, P. Hwang, A. Arai, A. T. Remaley, V. Sachdev, R. Shamburek, R. O. Cannon and E. D. Green, "The ClinSeq Project: piloting large-scale genome

sequencing for research in genomic medicine.", *Genome Res*, Vol. 19, No. 9, pp. 1665–1674, September 2009.

39. Firtina, C. and C. Alkan, "On genomic repeats and reproducibility.", *Bioinformatics*, March 2016.

40. Warren, R. L., G. G. Sutton, S. J. M. Jones and R. A. Holt, "Assembling millions of short DNA sequences using SSAKE", *Bioinformatics*, Vol. 23, No. 4, pp. 500–501, 2007.

41. Dohm, J. C., C. Lottaz, T. Borodina and H. Himmelbauer, "SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing", *Genome Research*, Vol. 17, No. 11, pp. 1697–1706, 2007.

42. Jeck, W. R., J. A. Reinhardt, D. A. Baltrus, M. T. Hickenbotham, V. Magrini, E. R. Mardis, J. L. Dangl and C. D. Jones, "Extending assembly of short DNA sequences to handle error", *Bioinformatics*, Vol. 23, No. 21, pp. 2942–2944, 2007.

43. Margulies, M. e. a., "Genome sequencing in microfabricated high-density picolitre reactors", *Nature*, Vol. 437, No. 15, pp. 376–380, September 2005.

44. Simpson, J. and R. Durbin, "Efficient de novo assembly of large genomes using compressed data structures", *Genome Research*, Vol. 22, pp. 549–556, December 2012.

45. Donmez, N. and M. Brudno, "Hapsembler: An Assembler for Highly Polymorphic Genomes", *Proceedings of the 15th Annual International Conference on Research in Computational Molecular Biology*, RECOMB'11, pp. 38–52, Springer-Verlag, Berlin, Heidelberg, 2011.

46. Chaisson, M. J., B. Brinza, Dumitru and P. A. Pevzner, "De novo fragment assembly with short mate-paired reads: Does the read length matter?", *Genome Research*, Vol. 19, No. 2, pp. 336–346, December 2008.

47. Simpson, J. T., K. Wong, J. S. D., J. E. Schein, S. J. Jones and I. Birol, "ABySS: A parallel assembler for short read sequence data", *Genome Research*, Vol. 19, No. 6, pp. 1117–1123, February 2009.

48. Zerbino, D. R. and E. Birney, "Velvet: algorithms for de novo short read assembly using de Bruijn graphs.", *Genome Res*, Vol. 18, No. 5, pp. 821–829, May 2008.

49. Bankevich, A., S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, A. V. Pyshkin, A. V. Sirotkin, N. Vyahhi, G. Tesler, A. M. A. and P. A. Pevzner, "SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing", *Journal of Computational Biology*, Vol. 19, No. 5, pp. 455–477, May 2012.

50. Butler, J., I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum and D. B. Jaffe, "ALLPATHS: De novo assembly of whole-genome shotgun microreads", *Genome Research*, Vol. 18, pp. 810–820, 2008.

51. Chikhi, R. and G. Rizk, "Space-Efficient and Exact de Bruijn Graph Representation Based on a Bloom Filter.", *WABI*, Vol. 7534 of *Lecture Notes in Computer Science*, pp. 236–248, Springer, 2012.

52. Karp, R. M., "Reducibility among combinatorial problems.", *Complexity of Computer Computations*, pp. 85–104, 1972.

53. Taylor, D. L., *PHAST (PHAGE assembly suite and tutorial), A web-based genome assembly teaching tool*, computational biology, Davidson College, May 2012.

54. Simpson, J. T. and R. Durbin, "Efficient construction of an assembly string graph using the FM-index", *Bioinformatics*, Vol. 26, pp. 367–373, 2010.

55. Hernandez, D., P. François, L. Farinelli, M. Osterås and J. Schrenzel, "De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer", *Genome Research*, Vol. 18, pp. 802–809, March 2008.

56. Hossain, M. S., N. Azimi and S. Skiena, "Crystallizing short-read assemblies around seeds", *BMC Bioinformatics*, Vol. 10, No. 1, pp. 1–12, January 2009.

57. Compeau, P. E. and P. A. Pevzner, *Genome Construction: A Puzzle with a Billion Pieces*, pp. 36–65, University of California, San Diego, 2011.

58. Compeau, P. E., P. A. Pevzner and G. Tesler, "How to apply de Bruijn graphs to genome assembly", *Nature Biotechnology*, Vol. 29, No. 11, pp. 987–991, November 2011.

59. Mulyukov, Z. and P. A. Pevzner, "Euler-Pcr: finishing experiments for repeat resolution", *Pacific Symposium on Biocomputing*, Vol. 19, No. 2, pp. 199–210, 2002.

60. Zerbino, D. R., *Using the Velvet de novo assembler for short-read sequencing technologies*, Vol. 11, pp. 1–12, John Wiley & Sons, Inc, September 2010.

61. Gnerrea, S., I. MacCalluma, D. Przybylskia, F. J. Ribeiroa, J. N. Burtona, B. J. Walkera, T. Sharpea, G. Halla, T. P. Sheaa, S. Sykesa, A. M. Berlina, D. Airda, M. Costelloa, R. Dazaa, L. Williamsa, R. Nicola, A. Gnirkea, C. Nusbauma, E. S. Lander and D. B. Jaffe, "High-quality draft assemblies of mammalian genomes from massively parallel sequence data", *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 108, pp. 1513–1518, January 2011.

62. MacCallum, I., S. Przybylski, Dariuszand Gnerre, J. Burton, I. Shlyakhter, A. Gnirke, J. Malek, K. McKernan, S. Ranade, T. P. Shea, L. Williams, S. Young, C. Nusbaum and D. B. Jaffe, "ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads", *Genome Biology*, Vol. 10, pp. 1–10, October 2009.

63. Li, R., H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang and J. Wang, "De novo assembly of human genomes with massively parallel short read sequencing", *Genome Research*, pp.

1–8, December 2009.

64. Luo, R., B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, J. Tang, G. Wu, H. Zhang, Y. Shi, Y. Liu, C. Yu, B. Wang, Y. Lu, C. Han, D. W. Cheung2, S.-M. Yiu, P. Shaoliang, Z. Xiaoqian, G. Liu, X. Liao, Y. Li, H. Yang, J. Wang, T.-W. Lam and J. Wang, "SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler", *Giga Science*, Vol. 1, No. 18, pp. 1–6, 2012.

65. Iqbal, Z., M. Caccamo, I. Turner, P. Flicek and G. McVean, "De novo assembly and genotyping of variants using colored de Bruijn graphs", *Nature Genetics*, Vol. 44, No. 2, pp. 226–232, February 2012.

66. Koren, S., M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis and A. M. Phillippy, "Hybrid error correction and de novo assembly of single-molecule sequencing reads", *Nature Biotechnology*, Vol. 30, pp. 693–700, May 2012.

67. Boisvert, S., F. Laviolette and J. Corbeil, "Ray: Simultaneous Assembly of Reads from a Mix of High-Throughput Sequencing Technologies", *Journal of Computational Biology*, Vol. 17, No. 11, pp. 1519–1533, November 2010.

68. Chevreux, B., T. Pfisterer and B. Drescher, "Using the miraEST Assembler for Reliable and Automated mRNA Transcript Assembly and SNP Detection in Sequenced ESTs", *Genome Research*, Vol. 14, No. 6, pp. 1147–1159, June 2004.

69. Deshpande, V., E. D. K. Fung, S. Pham and V. Bafna, "Cerulean: A Hybrid Assembly Using High Throughput Short and Long Reads", *WABI*, Vol. 8126, pp. 349–363, Springer-Verlag, Heidelberg, Berlin, 2013.

70. Au, K. F., J. G. Underwood, L. Lee and W. H. Wong, "Improving PacBio Long Read Accuracy by Short Read Alignment", *PLOS One*, Vol. 7, No. 10, pp. 1–8, October 2012.

71. Wang, Y., Y. Yu, B. Pan, P. Hao, Y. Li, Z. Shao, X. Xu and X. Li, "Optimizing hybrid assembly of next-generation sequence data from Enterococcus faecium: a microbe with highly divergent genome", *BMC Systems Biology*, Vol. 6, No. S-3, p. S21, 2012.

72. Chevreux, B., T. Wetter and S. Suhai, "Genome sequence assembly using trace signals and additional sequence information", *Computer Science and Biology:Proceedings of the German Conference on Bioinformatics (GCB)*, Vol. 99, pp. 45–56, 1999.

73. Ergüner, B., D. Üstek and M. Şamil Sağıroğlu, "Performance comparison of Next Generation sequencing platforms", *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6453–6456, August 2015.

74. Altschul, S., W. Gish, W. Miller, E. Myers and D. J. Lipman, "Basic local alignment search tool", *Journal of Molecular Biology*, Vol. 215, No. 3, pp. 403–410, October 1990.

75. Zhang, Z., S. Schwartz, L. Wagner and W. Miller, "A greedy algorithm for aligning DNA sequences", *Journal of Computational Biology*, Vol. 12, No. 7, pp. 203–214, 2000.

76. Sharp, A. J., Z. Cheng and E. E. Eichler, "Structural variation of the human genome", *Annu Rev Genomics Hum Genet*, Vol. 7, pp. 407–442, 2006.

77. Mills, R. E. e. a., "Mapping copy number variation by population-scale genome sequencing.", *Nature*, Vol. 470, No. 7332, pp. 59–65, February 2011.

78. The 1000 Genomes Project Consortium, "A global reference for human genetic variation.", *Nature*, Vol. 526, No. 7571, pp. 68–74, September 2015.

79. Medvedev, P., M. Stanciu and M. Brudno, "Computational methods for discover-

ing structural variation with next-generation sequencing.", *Nat Methods*, Vol. 6, No. 11 Suppl, pp. S13–S20, November 2009.

80. Church, D. M., V. A. Schneider, K. M. Steinberg, M. C. Schatz, A. R. Quinlan, C.-S. Chin, P. A. Kitts, B. Aken, G. T. Marth, M. M. Hoffman, J. Herrero, M. L. Z. Mendoza, R. Durbin and P. Flicek, "Extending reference assembly models.", *Genome Biol*, Vol. 16, p. 13, 2015.

81. Marschall, T. e. a., "Computational Pan-Genomics: Status, Promises and Challenges", *bioRxiv*, 2016.

82. Hajirasouliha, I., F. Hormozdiari, C. Alkan, J. M. Kidd, I. Birol, E. E. Eichler and S. C. Sahinalp, "Detection and characterization of novel sequence insertions using paired-end next-generation sequencing.", *Bioinformatics*, Vol. 26, No. 10, pp. 1277–1283, May 2010.

83. Ye, K., M. H. Schulz, Q. Long, R. Apweiler and Z. Ning, "A pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads", *Bioinformatics*, Vol. 25, No. 21, pp. 2865–2871, November 2009.

84. Iqbal, Z., M. Caccamo, I. Turner, P. Flicek and G. McVean, "De novo assembly and genotyping of variants using colored de Bruijn graphs", *Nature genetics*, Vol. 44, No. 2, pp. 226–232, 2012.

85. Xia, L. C., S. Sakshuwong, E. S. Hopmans, J. M. Bell, S. M. Grimes, D. O. Siegmund, H. P. Ji and N. R. Zhang, "A genome-wide approach for detecting novel insertion-deletion variants of mid-range size", *Nucleic Acids Research*, Vol. 44, No. 15, p. e126, June 2016.

86. Kidd, J. M., T. Graves, T. L. Newman, R. Fulton, H. S. Hayden, M. Malig, J. Kallicki, R. Kaul, R. K. Wilson and E. E. Eichler, "A human genome structural variation sequencing resource reveals insights into mutational mechanisms.", *Cell*,

Vol. 143, No. 5, pp. 837–847, November 2010.

87. Hach, F., F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler and S. C. Sahinalp, "mrsFAST: a cache-oblivious algorithm for short-read mapping.", *Nat Methods*, Vol. 7, No. 8, pp. 576–577, August 2010.

88. Hach, F., I. Sarrafi, F. Hormozdiari, C. Alkan, E. E. Eichler and S. C. Sahinalp, "mrsFAST-Ultra: a compact, SNP-aware mapper for high performance sequencing applications.", *Nucleic Acids Res*, Vol. 42, No. Web Server issue, pp. W494–W500, July 2014.

89. Bailey, J. A., A. M. Yavor, H. F. Massa, B. J. Trask and E. E. Eichler, "Segmental duplications: organization and impact within the current human genome project assembly.", *Genome Res*, Vol. 11, No. 6, pp. 1005–1017, June 2001.

90. Altschul, S. F., W. Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic local alignment search tool.", *J Mol Biol*, Vol. 215, No. 3, pp. 403–410, October 1990.

91. Karakoc, E., C. Alkan, B. J. O'Roak, M. Y. Dennis, L. Vives, K. Mark, M. J. Rieder, D. A. Nickerson and E. E. Eichler, "Detection of structural variants and indels within exome data.", *Nat Methods*, Vol. 9, No. 2, pp. 176–178, 2012.

92. Kahn, A. B., "Topological Sorting of Large Networks", *Commun. ACM*, Vol. 5, No. 11, pp. 558–562, November 1962.

93. Smith, T. F. and M. S. Waterman, "Identification of common molecular subsequences.", *J Mol Biol*, Vol. 147, No. 1, pp. 195–197, March 1981.

94. Johnson, D. S., "Approximation Algorithms for Combinatorial Problems", *J. Comput. Syst. Sci.*, Vol. 9, No. 3, pp. 256–278, December 1974.

95. Danecek, P., A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. Handsaker, G. Lunter, G. Marth, S. T. Sherry, G. McVean, R. Durbin and

T. G. P. A. Group, "The Variant Call Format and VCFtools.", *Bioinformatics*, June 2011.

96. "Pamir source code directory", `https://bitbucket.org/compbio/pamir`, accessed at January 2017.

97. Huang, W., L. Li, J. R. Myers and G. T. Marth, "ART: a next-generation sequencing read simulator.", *Bioinformatics*, Vol. 28, No. 4, pp. 593–594, February 2012.

98. Huddleston, J., M. J. Chaisson, K. M. Steinberg, W. Warren, K. Hoekzema, D. S. Gordon, T. A. Graves-Lindsay, K. M. Munson, Z. N. Kronenberg, L. Vives, P. Peluso, M. Boitano, C.-S. Chin, J. Korlach, R. K. Wilson and E. E. Eichler, "Discovery and genotyping of structural variation from long-read haploid genome sequence data", *Genome Research*, , No. Accepted, November 2016.

99. Steinberg, K. M., V. A. Schneider, T. A. Graves-Lindsay, R. S. Fulton, R. Agarwala, J. Huddleston, S. A. Shiryev, A. Morgulis, U. Surti, W. C. Warren, D. M. Church, E. E. Eichler and R. K. Wilson, "Single haplotype assembly of the human genome from a hydatidiform mole.", *Genome Res*, Vol. 24, No. 12, pp. 2066–2076, December 2014.

100. Kidd, J. M. e. a., "Mapping and sequencing of structural variation from eight human genomes.", *Nature*, Vol. 453, No. 7191, pp. 56–64, May 2008.

101. Zook, J. M. e. a., "Extensive sequencing of seven human genomes to characterize benchmark reference materials", *Scientific Data*, Vol. 160025, No. 3, p. online, June 2016.

102. Sherry ST, K. M. B. J. P. L. S. E. S. K., Ward MH, "dbSNP: the NCBI database of genetic variation", *Nucleic Acids Res*, Vol. 29, No. 1, pp. 308–11, 2001.