Title

Efficient Algorithms for Human Genetic Variation Detection using High-throughput Sequencing Techniques

Author

He, Dan

Publication Date

2012

Peer reviewed|Thesis/dissertation

University of California

Los Angeles

Efficient Algorithms for Human Genetic Variation Detection using High-throughput Sequencing Techniques

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Dan He

2012

Abstract of the Dissertation

Efficient Algorithms for Human Genetic Variation Detection using High-throughput Sequencing Techniques

by

Dan He

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2012

Professor Eleazar Eskin, Chair

High-throughput sequencing (HTS) technologies are one type of genome sequencing techniques where short DNA segments, or reads, are sequenced or sampled from genome. Compared with the traditional genome sequencing techniques, they have advantages such as low-cost and they are able to parallelize the sequencing process to produce millions of reads. These technologies have been widely used in many important problems related to human genetic variations. We mainly target three human genetic variation problems with the reads generated by HTS.

It is well-known that human individuals differ from each other by 0.1%. The majority of the differences is in the form of SNPs, or Single Nucleotide Polymophisms. Haplotypes, defined as the sequences of SNPs on each chromosome of a human genome, are important for problems such as imputation of genetic variants, relatedness of human individuals, etc. A difficulty in haplotype inference is the presence of sequencing errors and a natural formulation of the problem is to infer haplotypes which are most consistent with the data from a combinatorial

perspective. Unfortunately, this formulation of the haplotype assembly is known to be NP-hard. We proposed a few techniques including dynamic programming, MaxSAT and Hidden Markov Model (HMM) to solve the problem optimally from different perspectives.

Structural variations and in particular Copy Number Variations (CNV) have dramatic effects of disease and traits. We first proposed an efficient algorithm to detect and reconstruct CNVs in unique genomic regions, where the sequencing reads generated from HTS are mapped to a reference genome and signatures indicating the presence of a CNV are identified. Then we extend the algorithm to a much more challenging problem where CNVs are in repeat-rich regions and the reads may be mapped to multiple mapping positions. To our knowledge, our method is the first attempt to both identify and reconstruct CNVs in repeat-rich regions.

Recent advances in sequencing technologies set the stage for large population based studies, in which the DNA or RNA of thousands of individuals will be sequenced. A few multiplexing schemes have been suggested, in which a small number of DNA pools are sequenced, and the results are then deconvoluted using compressed sensing or similar approaches. These methods, however, are limited to the detection of rare variants. We provide a new algorithm for the deconvolution of DNA pools multiplexing schemes. The presented algorithm utilizes a likelihood model and linear programming and is able to genotype both low and high allele frequency SNPs with microarray genotyping and imputation.

The dissertation of Dan He is approved.

Amit Sahai

Stanley Nelson

Douglas Stott Parker

Eleazar Eskin, Committee Chair

University of California, Los Angeles

2012

To my Parents, my wife Vivian, and my son Dennis.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I express my sincerest gratitude to my dissertation advisor Dr. Eleazar Eskin, who has supported me thoughout my research with his patience and guidance while allowing me the room to work in my own way and to work on the topics I am interested in. I truly feel that I was very fortunate to learn research under his supervision and the research was so enjoyable and memorable. He taught me how to become a better researcher and guided me to produce influential research work. Without his encouragement and effort, this thesis would not have been completed or written.

I would like to thank my committee members: Dr. Amit Sahai, Dr. Douglas Stott Parker, and Dr. Stanley Nelson for their valuable time, comments, and encouragements on the research. I would also like to thank my mentor Nebojsa Jojic at Microsoft Research for his guidance on a project in a new field to me. Although I am not able to list everyone's name but I own supreme thanks to all of my collaborators and my lab peers for the inspiration they brought to me, the fruitful discussions we had and the amazing skills they shared with me.

Any words of thanks are too little for my parents and my parents-in-law for always being there for me with their supports and encouragements. When my son Dennis was born two months before my final defense, they visited us and helped us to take care of him. Without their help, I could not have finished the thesis in time. I would like to especially thank my wife, Vivian. Without her support and understanding, I would not have my years at UCLA so fruitful and enjoyable.

Vita

1982	Born
	Mianyang, Sichuan, China
2004	B.S. Computer Science
	University of Science and Technology of China
	Hefei, Anhui, China
2006	M.S. Computer Science
	University of Vermont
	Burlington, Vermont, USA
2006-2008	Software Engineer
	Plateau Systems LTD.
	Arlington, Virginia, USA
2011	Research Intern
	Microsoft Research
	Redmond, Washington, USA

Publications

Dan He, Buhm Han, Eleazar Eskin. "Optimal Algorithm for Haplotype Phasing with Imputation using Sequencing Data." In *Proceedings of the 16th Annual International Conference on Research in Computational Molecular Biology (Recomb 2012)*, accepted, Barcelona, Spain, 2012.

Dan He, Farhad Hormozdiari, Nicholas Furlott, Eleazar Eskin. "Efficient Algorithms for Tandem Copy Number Variation Reconstruction in Repeat-rich Regions." In *Bioinformatics*, 2011.

Dan He, Xingquan Zhu, Douglas S. Parker. "How Does Research Evolve? Pattern Mining for Research Meme Cycles." In *Proceedings of the 2011 IEEE International Conference on Data Mining (ICDM2011)*, Vancouver, Canada, 2011.

Dan He. "Mining Research Topic-related Influence between Academia and Industry." In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD2011)*, Athens, Greece, 2011.

Dan He, Arthur Choi, Knot Pipatsrisawat, Adnan Darwiche and Eleazar Eskin. "Optimal Algorithms for Haplotype Assembly From Whole-Genome Sequence Data." In *Proceedings of the 18th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2010)*, Boston, USA, 2010.

Dan He, Xindong Wu, Xingquan Zhu. "Approximate Repeating Pattern Mining with Gap Requirements." In *Jounral of Computational Intelligence*, 2010.

Dan He, Douglas S. Parker. "Topic Dynamics: an alternative model of 'Bursts' in Streams of Topics." In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD2010)*, Washington DC, USA, 2010.

CHAPTER 1

Introduction

High-throughput sequencing (HTS) technologies (also known as Next-Generation Sequencing) are one type of genome sequencing techniques where short DNA segments, or reads, are sequenced or sampled from genome. Compared with the traditional genome sequencing techniques, they have advantages such as low-cost and they are able to parallelize the sequencing process to produce millions of reads. These technologies have been widely used in many important problems related to human genetic variations. We mainly target three human genetic variation problems with the reads generated by HTS.

Chapter 2: Optimal Algorithms for Haplotype Assembly

It is well-known that human individuals differ from each other by 0.1%. The majority of the differences is in the form of SNPs, or Single Nucleotide Polymophisms. Haplotypes, defined as the sequences of SNPs on each chromosome of a human genome, are important for problems such as imputation of genetic variants, relatedness of human individuals, etc.

Unfortunately, current technology to collect genetic information is through whole-genome sequencing, where very short DNA segments are sampled, or sequenced, from the whole DNA sequence. Therefore it is only able to provide haplotype information for very short regions. The haplotype assembly problem

which attempts to combine the information from all of these regions to infer complete haplotypes is one of the most important problems in population genetics and has attracted lots of attention. A difficulty in haplotype inference is the presence of sequencing errors and a natural formulation of the problem is to infer haplotypes which are most consistent with the data from a combinatorial perspective. Unfortunately, this formulation of the haplotype assembly is known to be NP-hard. We proposed a few techniques to solve the problem effectively. We first developed a dynamic programming algorithm, which is able to optimally infer haplotypes when the DNA segments are short. Next we converted the problem into a MaxSAT problem which aims inferring haplotypes from longer DNA segments and we solved the MaxSAT problem using SAT solvers. Finally we combined dynamic programming with a Hidden Markov Model (HMM) to infer haplotypes using a set of reference DNA sequences, which is able to improve the inference accuracy significantly.

This chapter is based on [HCP10], presented at the ISMB 2010 conference, which is also presented at the journal of Bioinformatics, 2010 and [HHE12], accepted by the Recomb 2012 conference, which will be also presented at the journal of Computational Biology, 2012.

Chapter 3: Efficient Algorithms for Reconstruction of Tandem Copy Number Variation

Structural variations and in particular Copy Number Variations (CNV) have dramatic effects of disease and traits. Technologies for identifying CNVs have been an active area of research for over 10 years. The current generation of high-throughput sequencing techniques presents new opportunities for identification of CNVs. Methods that utilize these technologies map sequencing reads to a

reference genome and look for signatures which might indicate the presence of a CNV. These methods work well when CNVs lie within unique genomic regions. However, the problem of CNV identification and reconstruction becomes much more challenging when CNVs are in repeat-rich regions, due to the multiple mapping positions of the reads. We first proposed an efficient algorithm to detect and reconstruct CNVs in non-repeat regions. Then we extend the algorithm to handle the multi-mapping reads such that the CNVs can be reconstructed with high accuracy even for repeat-rich regions. To our knowledge, our method is the first attempt to both identify and reconstruct CNVs in repeat-rich regions.

This chapter is based on [HFE10] and [HHF11], presented at the GIW 2010 and HiTSeq 2011 conferences, respectively. The two works are also presented at the journal of BMC Bioinformatics, 2010 and Bioinformatics, 2011.

Chapter 4: Genotyping common and rare variation using overlapping pool sequencing

Recent advances in sequencing technologies set the stage for large population based studies, in which the DNA or RNA of thousands of individuals will be sequenced. However, such studies are still infeasible using a straightforward sequencing approach; as a result, recently a few multiplexing schemes have been suggested, in which a small number of DNA pools are sequenced, and the results are then deconvoluted using compressed sensing or similar approaches. These methods, however, are limited to the detection of rare variants. We provide a new algorithm for the deconvolution of DNA pools multiplexing schemes. The presented algorithm utilizes a likelihood model and linear programming. The approach allows for the addition of external data, particularly imputation data, resulting in a flexible environment that is suitable for different applications. Par-

ticularly, we demonstrate that both low and high allele frequency SNPs can be accurately genotyped when the DNA pooling scheme is performed in conjunction with microarray genotyping and imputation. Additionally, we demonstrate the use of our framework for the detection of cancer fusion genes from RNA sequences.

This chapter is based on [HZP11], presented at the Recomb-Seq 2011 workshop. This work is also presented at the journal of BMC Bioinformatics, 2011.

Chapter 5: Conclusion and Future Work

We first summarize our contribution to the human genetic variation related problems using HTS. Our future work will be focusing on developing methods to improve IBD (identity-by-descent) estimation in multiple individuals and pedigree reconstruction for a group of individuals based on their haplotypes.

CHAPTER 2

Optimal Algorithms for Haplotype Assembly

2.1 Haplotype Assembly from Whole-genome Sequencing Data

2.1.1 Introduction

Obtaining haplotypes, or the sequence of alleles on each chromosome, is an important step for many types of analyses of genetic variation in the human genomes. In particular, haplotype inference is required for the application of many imputation algorithms [MHM07b] which are now widely applied in the analysis of genome-wide association studies.

The standard approach for obtaining haplotype information involves collecting genotype data from a population of individuals. Genotype data contains information on the set of alleles at each locus, but lacks information on which chromosome a particular allele occurs on. Computational methods are then applied to these genotype data to infer the haplotypes [SSD01, HE04, BB08b]. These methods take advantage of the fact that alleles at neighboring loci in the genomes are correlated or are "in linkage disequilibrium" (LD), as well as the fact that in any given region, only a few common haplotypes account for the majority of the genetic variations in the population. Because of their reliance

on LD, these methods have difficulty inferring haplotypes with rare variants and have no ability to infer haplotypes for alleles that are unique to an individual.

Recently, the development of high-throughput sequencing technology has enabled an alternative strategy to obtain haplotypes. Since each sequence read is from a single chromosome, if a read covers two variant sites, all of the alleles present in the read must be from the same haplotype. Using this insight, it is possible to assemble the two haplotypes for a chromosome from the collection of such reads by joining reads which share alleles at common variants. The problem is referred to as "haplotype assembly" [LBI01], which is challenging in the following two aspects:

- Reads are sampled from either of the two haplotypes and no information is given about which one they come from. The reads need to be separated for the two haplotypes in the assembly process.

- Errors in reads significantly increase the difficulty of the problem and it has been shown that the problem is NP-hard even for reads of length 2 [LBI01, CIK05].

A simple greedy heuristic method [LSN07] (which we call the *Greedy* algorithm), concatenates the reads with minimum conflicts and is fast but not very accurate when reads contain errors. Other stochastic algorithms, such as HASH [BHA08], which is a Markov Chain Monte Carlo (MCMC) algorithm, and HapCut [BB08a], which is a combinatorial approach, have been shown to be much more accurate than the Greedy algorithm on the HuRef diploid genome sequence [LSN07].

However, both HASH and HapCut algorithms use stochastic strategies and therefore are not guaranteed to find optimal solutions for the haplotype assembly problem. We propose a dynamic programming algorithm, which is able to assem-

ble the haplotypes optimally with time complexity $O(m \times 2^k \times n)$, where m is the number of reads, k is the length of the longest read and n is the total number of heterozygous sites in the haplotypes. Since this time complexity is exponential in k, we reduce the problem to the maximum satisfiability (MaxSAT) problem for cases where k is large. MaxSAT conversion is a well-known strategy for many computational biology problem such as SNP Tagging [CZH08]. The converted MaxSAT problem can often be solved optimally in a reasonable amount of time with a MaxSAT solver. Our experiments show that the MaxSAT approach can solve 99.98% instances of the converted haplotype assembly problem optimally. We also show for the first time that the current best-known solution is only 1.1% from the optimal solution and our solution is the best result that has yet been achieved.

Taking advantage of the efficiency and optimality of our method, we are able to perform simulation experiments to evalute the feasibility of assembling haplotypes using sequence reads with the length typical of the current high-throughput technologies. The current sequencing technologies are able to collect paired-end reads where sequences of two segments are obtained separated by an approximate distance (insert length). Our experiments show that the insert length and in particular the variability in the insert length play a crucial role in our ability to assemble haplotypes. Using data from HapMap [Int07] we demonstrate that using current high-throughput sequencing technologies, the assembly of reads into haplotypes is impractical. However, we show that combining haplotype assembly from sequencing with traditional approaches for inferring haplotypes using genotypes can effectively recover haplotypes for both common and rare alleles.

2.1.2 Related Work

The haplotype assembly problem was first introduced by [LBI01]. They show that the problem is computationally challenging when reads contain errors since the reads can not be partitioned perfectly into two disjoint sets. Therefore, various combinatorial objective functions have been proposed [LBI01, LSL02] to define the best reconstruction of haplotypes such as minimum fragment removal (MFR), minimum error correction (MEC), minimum SNP removal (MSR), minimum implicit SNP removal (MISR), minimum implicit fragment removal (MIFR). Out of these objective functions, MEC which is the number of conflicts between the sequence reads and the constructed haplotypes, is the most difficult one to optimize. The haplotype assembly problem with MEC as the object function is NP-hard even for gapless reads of length 2, while polynomial algorithms exist for solving the problem with MFR and MSR as the objective function [LSL02, CIK05]. Several heuristic and stochastic methods [PS04, WWL05, LSN07, BHA08, BB08a] have been proposed to optimize MEC for gapped reads. We also focus on minimizing MEC. Therefore, the "haplotype assembly" problem can be defined as following: given a set of reads which may contain errors, reconstruct the pair of haplotypes by partitioning the reads to either haplotype such that the Minimum Error Correction (MEC) is minimized.

The Greedy heuristic algorithm [LSN07], which concatenates the reads with minimum conflicts, is able to construct optimal haplotypes very quickly if the reads are error-free. When there are errors in the reads, the Greedy algorithm usually outputs much worse results than the optimal solution. HASH [BHA08] and HapCut [BB08a] algorithms are both based on the idea of building a graph from the read matrix where each row corresponds to a read and each column corresponds to a position of the haplotype. In the graph, each column is a node

and an edge between two nodes is created if there is a read spanning the corresponding two columns. The weights of the edges are determined by the number of reads that are consistent with the haplotypes minus the number of reads that are in conflict with the haplotypes in the two columns. The HASH algorithm uses graph cut computations to construct the Markov Chain used for sampling the haplotype space. HapCut uses Max-Cut computations in an associated graph to greedily move towards the optimal MEC solution. Both HASH and HapCut algorithms obtain much more accurate haplotypes than the Greedy algorithm. Since convergence of Markov Chain is slow, HapCut is much faster than HASH with almost the same accuracy.

2.1.3 Methods

We followed the notation by [BB08a] for the haplotype assembly problem. Given a reference genome sequence and the set of reads containing sequence from both chromosome, after aligning all the reads to the reference genome [LRD08a], the homozygous sites (columns in the alignment with identical values) are discarded since they are not informative. The heterozygous sites (columns in the alignment with different values) correspond to alleles which differ between chromosomes and they are labelled as 0 or 1 arbitrarily. A matrix X of size $m \times n$ can be built from the alignment, where m is the number of reads and n is the number of heterozygous sites. The i-th read is described as a ternary string $X_i \in \{0, 1, -\}^n$, where '$-$' indicates a gap, namely that the allele is not covered by the fragment (again following the notation of [BB08a] for clarity). The *start position* and *end position* of a read are the first and last positions in the corresponding row that are not '$-$', respectively. Therefore the '$-$'s in the head and tail of each row will not be considered as part of the corresponding read. However, there can be '$-$'s

inside each read which correspond to either missing data for single reads or gaps connecting a pair of single reads (called *paired end reads*). Reads without "−" are called *gapless reads*; otherwise they are called *gapped reads*. Assuming a read's end position is j, start position is i, the *length* of the read is defined as $j - i + 1$. We also assume all the reads have already been correctly aligned to the reference genome by some mapper, which may not be true since the mapper may introduce mapping errors and the reads may come from repeat-rich regions. However, the mapping process is out of the scope of this paper and we thus do not evaluate the effects of mapping errors on the quality of our haplotype assembly solution.

The haplotypes can be represented as an unordered pair of binary strings $H = (h_1, h_2)$, each of length n. Since all the sites are heterozygous, h_2 is the bit-wise complement of h_1. An example of the read matrix is shown in Table 2.1. As we can see in this example, each read corresponds to one row where '−' indicates missing information. Reads often contain errors. For example, if we only consider reads 1,2,3, we can partition them perfectly into two sets ({read1, read3}, {read2}) and re-construct the haplotypes as H=(h_1 = {0000}, h_2 = {1111}) by assigning reads 1 and 3 to h_1 and assigning read2 to h_2. However, read4 is in conflict with this partition and there is no perfect partition for reads 1,2,3,4.

Therefore, in the presence of errors, we need to reconstruct the haplotypes such that some objective function is minimized. The objective function we use is Minimum Error Correction (MEC), which is the minimum number of changes, or corrections, that need to be made in the read matrix such that the resulting matrix admits a perfect bi-partition, where each corrected read maps to either haplotype perfectly. Alternatively speaking, for any pair of complementary haplotypes, the set of reads can be partitioned into two subsets which satisfy the

following property: If both subsets of the reads are mapped to the two haplotypes at their corresponding intervals indicated by their starting positions and lengths, where one subset is mapped to only one of the haplotypes, the number of errors, or mismatches for the mapping is minimized. This minimum number of errors is the MEC score of the reads for the pair of haplotypes. In our example, if we only consider reads 1,2,3,4, we can change read4 from (101) to (001) such that now we can obtain a perfect bi-partition ({read1, read3, read4}, {read2}) with reconstructed haplotypes $H' = (h_1 = \{00001\}, h_2 = \{11110\})$. The number of changes we made is obviously 1. Therefore the "haplotype assembly" problem is identical to finding a pair of haplotypes H such that the MEC score of the reads in the read matrix is minimized. For example, the MEC score for reads 1,2,3,4 is 1 and the corresponding optimal pair of haplotypes is H'. This example is very simple, however, in reality, the number of reads can be very large and it has been shown that the "haplotype assembly" problem is NP-hard even for gapless reads of length 2.

Notice that the optimal haplotypes which minimize the MEC score may not be exactly the same as the real haplotypes. However, our objective function for the haplotype assembly problem makes the maximal parsimony assumption common in many computational biology problems [Fit77, Gus03]. Therefore the optimal solution is the most biologically meaningful solution for our problem. Another factor that may affect the quality of the reconstructed haplotypes is sequencing error. If the errors are consistent across reads, the reconstructed haplotypes may maintain these errors and may be incorrect. However, the MEC criteria attempts to discover haplotypes that minimizes the number of errors in the reconstruction. This is because if the errors are contained in only the minority of the reads, by minimizing the possible errors, the reconstructed haplotypes can still capture the reads that were sequenced correctly and thus avoid the sequencing errors.

reads	0	1	2	3	4	5	6	7	8	9	10	11	12
read1	0	0	-	-	-	-	-	-	-	-	-	-	-
read2	-	-	1	1	-	-	-	-	-	-	-	-	-
read3	0	0	0	0	-	-	-	-	-	-	-	-	-
read4	-	-	1	0	1	-	-	-	-	-	-	-	-
read5	-	-	0	-	-	0	-	-	-	-	-	-	-
read6	-	-	-	0	-	-	-	-	-	-	1	1	-
read7	-	-	-	-	0	0	0	-	-	-	-	-	-
read8	-	-	-	-	0	1	1	0	-	-	-	-	-
read9	-	-	-	-	-	-	-	-	1	1	-	-	-
read10	-	-	-	-	-	-	-	1	1	0	-	-	0

Table 2.1: An example of read matrix which consists of 10 reads spanning 13 positions.

2.1.3.1 Dynamic Programming Algorithm

To obtain the optimal solution for the haplotype assembly problem, a naive approach is to enumerate all binary strings, each of which represents a possible haplotype, and then assign the reads to each pair of possible haplotypes to minimize the conflicts with the reads. Given the length of haplotypes as n, the number of reads as m, this naive approach requires $O(m \times 2^n)$ complexity and is therefore infeasible for large n. However, the problem can be solved optimally using a dynamic programming algorithm as we show here. The basic idea of the dynamic programming algorithm is to store the optimal MEC for partial haplotypes (the prefixes for full-length haplotypes) ending with every possible length k binary strings. Then the algorithm extends the partial haplotypes by one bit repeatedly until full-length haplotypes are obtained.

We define r as a length k binary string and $s(i, r)$ as the MEC score for partial haplotypes starting at position 0 and ending at position $i + k - 1$ with suffix r where the partial haplotypes are the prefixes of full length haplotypes. $s(i, r)$ is obtained by considering only the reads whose starting positions are no greater than i and it solves a subproblem of the full length haplotype assembly where all reads are considered. We build a dynamic programming matrix and at each position i we store $s(i, r)$ for all r. The best MEC is the minimum $s(n - k, r)$ over all r, where n is the full length of the haplotypes. Given the definition of MEC (minimum number of changes (flips) needed), we can initialize $s(0, r)$ by considering the reads that start at position 0, and for each read compute the number of mismatches between the read and r and the read and the complement of r. The partial haplotypes at position i can be obtained by extending the partial haplotypes at position $i - 1$ with either a 0 or 1. $s(i, r_1)$ can be computed from $s(i - 1, r_2)$ and the newly introduced errors between r_1 and all reads starting at position i, where the length $k-1$ suffix of r_2 is the same as the length $k-1$ prefix of r_1. The recursion is illustrated in the following formula:

$$s(i, r) = \min_{b=0,1}(s(i - 1, (b, r[0, k - 2])) + h(i, r)) \tag{2.1}$$

where b is a binary bit of either 0 or 1, $r[0, k - 2]$ is the length $k - 1$ prefix of r, $(b, r[0, k - 2])$ is a length-k binary string generated by concatenating b with $r[0, k - 2]$, $h(i, r)$ is the minimum of the total number of disagreements between r or the complement of r and all reads starting at position i. Notice that, for $h(i, r)$, we consider both r and the complement of r because each read can be assigned to either the current haplotype or its complement (depending on which assignment produces smaller disagreements). The assignments producing the minimum disagreements are then selected. Each '$-$' matches both 0 and 1, so a

mismatch only happens between non-'$-$' symbols.

Starting from the solution which leads to a minimal value of $s(n-k, r)$ over all r, we can trace back the dynamic programming matrix to re-construct the haplotypes that minimize the MEC score. The time complexity of the dynamic programming algorithm is $O(m \times 2^k \times n)$, where m is the number of reads, k is the length of the longest read and n is the total number of SNPs in the haplotypes. Here, for illustrative purpose, we assume all reads are of length k. In reality the reads are of different length and the time complexity of the dynamic programming algorithm becomes $O(m \times 2^{k_{\max}} \times n)$, where $k_{\max}$ is the maximal number of alleles contained among all reads.

We also observe that we can split up the reads into sets where there is no read that spans any two sets. We call such a set a *block*. The set of reads can thus be partitioned into many blocks. Since no read spans any two blocks, which means those blocks are independent, we can reconstruct haplotypes for each block in parallel using the dynamic programming algorithm developed above and then concatenate the solutions for each block to construct the complete haplotypes.

Next we show a simple example for the dynamic programming algorithm. We take read1-read4 from Figure 2.1 as an example. We need to order them according to their start positions.

1. At position $i = 0$: We have read1 and read3, k = length(read3) = 4. Therefore we compute $s(i, r)$ for all length $k = 4$ binary strings r, using only read1 and read3:

   s(0, 0000) = h(0, 0000) = 0,

   s(0, 0001) = h(0, 0001) = 1,

   s(0, 0010) = h(0, 0010) = 1,

   ...,

s(0, 1111) = h(0, 1111) = 0.

2. At position $i = 1$: There is no read starting at position 1.

3. At position $i = 2$: We have read2 and read4 and $k = \text{length(read4)} = 3$. Again, we compute $s(i, r)$ for all length $k = 3$ binary strings r, using only read2 and read4. Since we do not have read starting at position 1, to simplify the computation, in Formula 1, $s(i-1, (b, r[0, k-2]))$ becomes $s(i-2, (b1, b2, r[0, k-2])$ where $b_1 \in \{0,1\}$ and $b_2 \in \{0,1\}$ are single binary bits. Therefore we have:

s(2, 000)=min(s(0, 0000)+h(2,000), s(0, 0100)+h(2,000), s(0, 1100)+h(2,000),
s(0, 1000)+h(2,000))=1,

s(2, 001)=min(s(0, 0000)+h(2,001), s(0, 0100)+h(2,001), s(0, 1100)+h(2,001),
s(0, 1000)+h(2,001))=1,

...,

s(2, 111)=min(s(0, 0011)+h(2,111), s(0, 0111)+h(2,111), s(0, 1111)+h(2,111),
s(0, 1011)+h(2,111))=1.

Therefore, the optimal MEC is $min(s(2, r))$ for all length $k = 3$ binary strings r and the optimal MEC is 1.

Notice that when we trace back to obtain the optimal haplotypes, it is not necessarily the case that there is only one pair of optimal haplotypes. In the above example, s(2, 000)=1 and it is from s(0, 0000). By tracing back from s(2, 000) we get the optimal haplotype pair ({00000}, {11111}). We also have s(2, 001)=1 and it is from s(0, 0000). By tracking back from s(2, 001) we get the optimal haplotype pair ({00001, 11110}). Both of the two pairs have optimal MEC of 1. When there are multiple optimal solutions, data on multiple individuals may be used to infer the most likely haplotypes for each ambiguous individual.

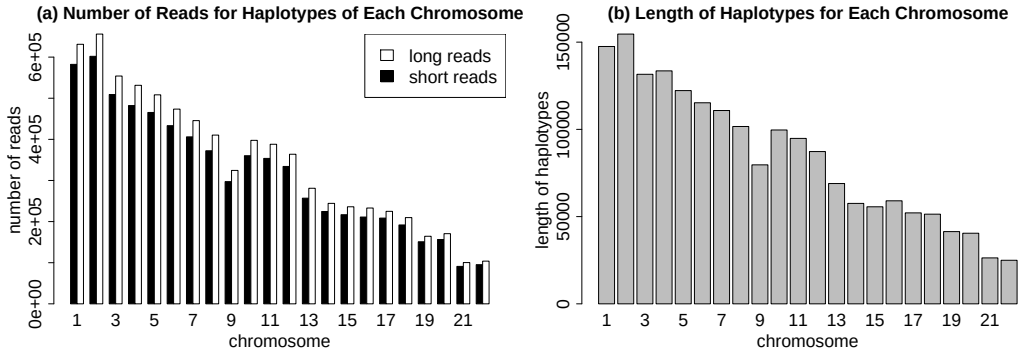

Figure 2.1: (a) The number of short reads, all reads and the length of haplotypes for each chromosome. The threshold for short reads is 15. (b) The length of haplotypes is the number of heterozygous sites in each chromosome.

2.1.3.2 MaxSAT Conversion for Haplotype Assembly

So far we only discussed a dynamic programming algorithm for single reads. Consider reads 5, 6 and 10 in Figure 2.1. In each of these reads, there are two continuous strings connected by "−", which indicates missing information. These reads are called *paired end reads* and are generated by modern sequencing technologies. The problem becomes much more complicated when paired end reads are considered since paired end reads usually span a long fragment, which can be as long as a few hundred positions. Although paired-end read can be considered as a special case of a single read, the dynamic programming algorithm introduced above becomes impractical, since we need to enumerate all positions the paired end read covers. As concluded above, the time complexity of the dynamic programming algorithm is $O(m \times 2^{k_{\max}} \times n)$ where $k_{\max}$ is the maximum number of alleles contained among all reads. When paired end reads are considered, $k_{\max}$ could be as large as a few hundred, making the dynamic programming approach impractical. Even single reads can be too long to enumerate some of the positions. We set a threshold for $k_{\max}$ such that the enumeration of all length $k_{\max}$

binary strings is computational feasible. We call the single reads and the paired end reads of length greater than the threshold *long reads* and the other reads as *short reads*.

We solve the haplotype assembly problem when long reads are also considered by conversion to MaxSAT. The *maximum satisfiability* problem (MaxSAT) is an optimization version of the well-known Boolean satisfiability problem (SAT) [BHM09]. Given a set of clauses (a clause is a disjunction of boolean literals), the MaxSAT problem asks for a complete assignment of all variables that maximizes the number of clauses the assignment satisfies. For example, consider the following set of four clauses:

$$(x_1), (\neg x_1 \vee x_2), (\neg x_1 \vee x_3), (\neg x_2 \vee \neg x_3)$$

The assignment $x_1 = \mathsf{false},\ x_2 = \mathsf{false},\ x_3 = \mathsf{false}$ satisfies three clauses and is optimal. We consider a variant of MaxSAT known as *partial MaxSAT*. Partial MaxSAT allows some clauses to be labelled as *hard*—i.e., their satisfiability is mandatory in any solution. The objective of the problem is to find an assignment that satisfies *all* hard clauses and satisfies the most number of non-hard (i.e, *soft*) clauses. For more discussion on MaxSAT and its variations, please see [LM09].

In our conversion of the haplotype assembly problem to partial MaxSAT, we define the following boolean variables:

- $h_i, 0 \leq i < n$, represents the binary symbol at position i in the haplotype to be constructed.
- $r_j, 0 \leq j < m$, represents the assignment of read j to a haplotype. The value $r_j = 0$ indicates that read j is assigned to the considered haplotype, while the value $r_j = 1$ indicates that the read is assigned the complement haplotype.

- $e_{ij}, 0 \leq i < n, 0 \leq j < m$, represents whether a correction is needed for position i of read j with respect to the considered haplotype. The value $e_{ij} = 1$ indicates that a correction is needed, while the value $e_{ij} = 0$ indicates that no correction is needed at that position.

Given these variables, we can define the set of clauses that describes the relationship between h_i, r_j, and e_{ij}. These clauses essentially specify that there is an error whenever the value at position i of read j does not match with the value at position i of the haplotype that the read is assigned to. Let $read[i][j]$ represent the value at position i of read j (i.e., the value of cell$[i][j]$ of the read matrix). We can formally define a set of clauses for each non-"$-$" entry in the read matrix as follows:

$$(h_i \Leftrightarrow \neg r_j \Leftrightarrow e_{ij}), \qquad \text{if } read[i][j] \text{ is } 0,$$
$$(h_i \Leftrightarrow r_j \Leftrightarrow e_{ij}), \qquad \text{if } read[i][j] \text{ is } 1.$$

Note that $\Leftrightarrow$ is the logical equivalence operator and that $(x \Leftrightarrow y \Leftrightarrow z)$ is a shorthand notation for the clauses $(x \vee y \vee z), (\neg x \vee \neg y \vee z), (\neg x \vee y \vee \neg z), (x \vee \neg y \vee \neg z)$. The above clause definition can be understood as follows. If read$[i][j]$ is 0, then the error e_{ij} is defined to be $h_i \Leftrightarrow \neg r_j$. That is, there should be an error if (and only if) (i) h_i is 1 and the read is assigned to the considered haplotype or (ii) h_i is 0 and the read is assigned to the complement haplotype. The case when read$[i][j]$ is 1 can be understood in a similar way. As these clauses describe how the errors are calculated, they should be hard clauses in our MaxSAT problem (they should not be violated by any solution).

Since we would like to find an assignment that minimizes error, we add to our MaxSAT problem the unit clause $(\neg e_{ij})$ for each non-"$-$" position in the read matrix. Every unit clause that an assignment falsifies (i.e., every error introduced)

will incur a penalty of 1 to that assignment. These unit clauses are soft clauses that might be falsified by the optimal solution. The optimal solution to this MaxSAT problem is simply any assignment that respects the error calculation rules and introduces the least amount of error.

This concludes our conversion of the haplotype assembly problem to partial MaxSAT problem. We may use any partial MaxSAT solver to solve the resulting problem. We used the solvers called Clone [PPC08] and WBO [MMP09] to solve the resulting MaxSAT problems.

2.1.4 HuRef Experiments

We first examine the performance of the dynamic programming algorithm on the filtered HuRef data from [LSN07] over all 22 chromosomes and directly compare our method to previous approaches [BB08a]. The data consists of 32 million DNA fragments generated by Sanger sequencing and contains a total of 1.85 million heterozygous variants for the 22 chromosomes. We show the number of short reads, all reads and number of heterozygous sites for each chromosome where the threshold for short reads is 15 in Figure 2.1. As we can see, the number of reads for each chromosome is very large. More than 90% of the reads are short reads. Haplotypes for each chromosome are also very long, making the haplotype assembly problem computationally intensive. The average number of reads that span each heterozygous site is between 6 and 7.

The whole-genome sequence data consists of many disconnected blocks where no read spans the boundary of two blocks. Therefore we can split the sequence data independently into many blocks and then solve the haplotype assembly problem for each block. The global MEC score is the sum of the scores from each block and the optimal haplotypes are the concatenation of the haplotypes from

each block. We show the read matrix for the first block of chromosome 22 in Figure 2.2 as an example, where we have around 2300 reads spanning a block of length around 400.

2.1.4.1 HuRef Experiments on Only Short Reads

We first compare the results of the dynamic programming algorithm with the results of Greedy and HapCut, on short reads, namely single reads and the paired end reads of length less than 15 only. As we showed in Figure 2.1, most of the reads are very short. However, there are still tens of thousands of reads of length more than 15. For example, in the block shown in Figure 2.2, there are around 400 long reads and the maximal length of the reads is around 200. We run all three algorithms on short reads only and the results are shown in Table 2.2. As we can see on average HapCut improves the MEC score of Greedy by 30%, while our dynamic programming algorithm shows for the first time that the solution from HapCut is only 1.1% from the optimal solution. The run time of the dynamic programming algorithm is reasonably fast and comparable to the HapCut algorithm. For example, for the first block of chromosome 22 in Figure 2.2, HapCut runs for 20 seconds while the dynamic programming algorithm runs for 24 seconds. The run times are even closer for small size blocks and around 90% blocks are such small size blocks. Both algorithms run for more than 10 hours and finish in roughly the same time on a computational cluster for all 22 chromosomes.

2.1.4.2 HuRef Experiments on All Reads

As mentioned in the previous section, in order to solve the haplotype assembly problem containing reads of all lengths, we convert the problem into a partial

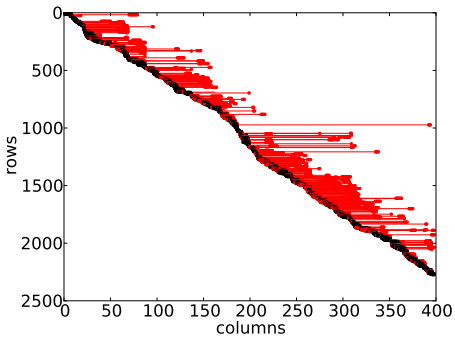

Figure 2.2: Graphical representation of the read matrix for the first block of Chromosome 22, where the reads are sorted by their starting positions. The rows are the reads and the columns are the haplotype positions. The black dots are the non-'−' cells for the short reads and the red dots are the non-'−' cells for the long reads. The red lines are the gap cells of the paired-end reads.

MaxSAT problem and use partial MaxSAT solvers to solve it. We consider two MaxSAT solvers: Clone [PPC08] and WBO [MMP09].

Out of 47,758 blocks, Clone was able to solve all but 8 blocks optimally. WBO, on the other hand, solved all but 34 blocks optimally. These two solvers report the same optimal solutions for the blocks they both solved. Interestingly, even for the 8 blocks that Clone could not solve optimally, it still reported solutions with lower MEC scores than those obtained from the HapCut algorithm.[1] We compared our results with the results of Greedy and HapCut. The results are shown in Table 2.2. As we can see, on average, HapCut improves the MEC score of Greedy by 34%, while our MaxSAT conversion method again shows that the solution of HapCut is very close to the optimal solution. Although there are 8 blocks that we could not solve optimally, the remaining 99.98% of the blocks were optimally solved. Therefore it is reasonable to believe that the overall solution

[1]Clone is an any-time algorithm that reports the current best solution as soon as it is discovered.

we obtained is very close to optimal. The run time of the MaxSAT solver on our cluster machine is around 15 hours, which is comparable to that of HapCut.

2.1.5 Designing Haplotype Assembly Protocols

While previously developed haplotype assembly approaches have been successfully applied to the HuRef data, it is not clear how applicable these approaches would be using current high-throughput genotyping technologies which have much shorter read lengths, yet higher coverage than the HuRef data. We take advantage of the efficiency of our algorithm to perform simulations in order to design sequencing protocols for current high-throughput technology in order to effectively obtain haplotypes. Unlike the reads for HuRef data, which are sequenced with the Sanger-based whole-genome shotgun sequencing and therefore are very long (each segment is thousands of base-pairs long including both homozygous and heterozygous sites), here we consider the reads generated by the high-throughput sequencing (HTS) technology [WSE08]. The reads generated by HTS are usually very short (each segment is around 30-100 base-pairs including both homozygous and heterozygous sites).

The basic parameters of sequencing technology that we explore are the sequence coverage ratio (the number of times that each base-pair in the sequence is covered), the insert length of the paired-end reads (the distance between the two segments of the paired-end reads), the variance of this insert length and the read length. We explore how these parameters affect the haplotype assembly. We perform our experiments over individual genotype data from HapMap [Int07]. For a single individual, we concatenate the heterozygous SNPs to construct a true haplotype. For the individual we downloaded, there are 505,065 heterozygous SNPs. We then mimic the sequencing process by randomly generating paired-end reads

	On Short Reads			On All Reads		
	Greedy	*HapCut*	*DP*	*Greedy*	*HapCut*	*MaxSAT*
*Chromosome*1	21355	15312	15292	29518	19687	19584
*Chromosome*2	16067	11251	11107	22706	14615	14576
*Chromosome*3	11909	8223	8181	16696	10702	10647
*Chromosome*4	12518	8820	8775	17509	11525	11304
*Chromosome*5	11621	8017	7944	16432	10536	10528
*Chromosome*6	10624	7487	7369	15295	9842	9826
*Chromosome*7	11668	8531	8423	17188	11244	11187
*Chromosome*8	10501	7343	7311	14535	9741	9025
*Chromosome*9	10199	7350	7312	13512	9222	9201
*Chromosome*10	10263	7313	7236	15076	9846	9778
*Chromosome*11	8825	6224	6196	12667	8200	8183
*Chromosome*12	8641	6337	6155	12453	8218	8176
*Chromosome*13	6412	4396	4341	8848	5822	5761
*Chromosome*14	6634	4567	4532	9070	5879	5845
*Chromosome*15	9289	6653	6623	13291	9311	9285
*Chromosome*16	8574	6160	6093	12365	8259	8207
*Chromosome*17	7088	5034	4955	10195	6525	6459
*Chromosome*18	4973	3526	3398	8324	4991	4943
*Chromosome*19	5549	3996	3907	7939	5319	5288
*Chromosome*20	4136	2909	2891	5563	3739	3723
*Chromosome*21	3877	2903	2796	5607	3888	3881
*Chromosome*22	4424	3267	3250	6685	4495	4479
Sum	**205147**	**145619**	**144087**	**291474**	**191606**	**189886**

Table 2.2: The MEC scores computed by Greedy, HapCut and dynamic programming, MaxSAT conversion, on short reads only and on all reads, respectively, for each chromosome.

with varying parameters including coverage ratio, insert length of the reads and standard deviation of the insert length. The insert length follows a Gaussian distribution with mean 1000. Assume the genome length is n, the sequence read length is l, the coverage ratio is c, the number of reads to be generated then is $\frac{n \times c}{l}$. The starting positions of the reads are randomly selected within the range of the whole genome, therefore they may cover both heterozygous and homozygous SNPs. The segment length of the sequence paired-end read is 36 (including both heterozygous sites and homozygous sites) which is a reasonable value given current technology. To evaluate the effects of these parameters on the assembly process, we first divide the haplotypes into blocks with distances greater than one standard deviation above the sum of the mean of the insert length (we use 1000). The reads are then very unlikely to span two blocks due to the distance between them. The length of a read in the read matrix is the number of heterozygous SNPs the read covers. Since we generate only paired-end reads in our simulation, which consist of two segments, if only one segment of a read covers heterozygous SNPs, the resulting read in the read matrix will be considered as a single read, otherwise it is considered as a paired-end read. The insertion of a paired-end read corresponds to the gap of the read in the read matrix. Although the mean of the insert length is 1000, the corresponding gap length in the read matrix is very small because only heterozygous SNPs are considered for assembly. Therefore, almost all the reads are very short. To illustrate this, we vary the coverage ratio as 10, 20, 30, 40 times, the standard deviation of insert length as 5, 50, 500. For all combinations of parameter settings, the ratios of short reads, namely reads of length less than 15, out of all reads, are all greater than 99.99%, indicating that our dynamic programming algorithm is indeed very practical and can be considered as optimal.

To evaluate how well the haplotype assembly can be done w.r.t the sequenc-

	(10, 5)	(10, 50)	(10, 500)	(20, 5)	(20, 50)	(20, 500)	(30, 5)	(30, 50)	(30, 500)
Num	8	7	10	8	6	6	8	6	4
$Size$	2	2	4	2	3	8	2	3	13
	(40, 5)	(40, 50)	(40, 500)	(100, 500)					
Num	8	5	3	1					
$Size$	2	4	17	31					

Table 2.3: Average number of connected components contained in each block and average size of the connected components whose size is greater than 1 for different (coverage ratio, standard deviation) settings.

ing protocols, we next construct a graph from the read matrix. Each heterozygous SNP is a vertex in the graph and we draw an edge between two vertices if their corresponding SNPs are covered by the same read. We construct such a graph using all the generated reads and consider the connected components in this graph since we have no information on how to phase heterozygous sites in different connected components relative to each other. The number of optimal solutions will be exponential in the number of connected components. Therefore, the smaller the number of connected components is, the better we can assemble the haplotypes. We count the average number of connected components each block contains. We also compute the average size of the connected components. We show the experimental results in Table 2.3. As we can see, the number of connected components in each block decreases as coverage ratio increases and as standard deviation increases. Meanwhile, the average size of connected components also increases. However, to reduce the number of connected components each block contains to one such that we can fully reconstruct each block, we need to use a very high coverage ratio such as 100. Thus for any reasonable coverage ratio that would be collected in a sequencing study, haplotype assembly will not be able to assemble haplotypes because there will not be enough reads to connect all of the variants into complete haplotypes.

However, the strategy of haplotype assembly can be combined with traditional haplotype inference techniques [SSD01, HE04] to infer haplotypes. The basic idea is that genotypes are obtained from the sequence data by performing SNP calling [LRD08a] in the sequence reads. The majority of the common variants will be present in the reference datasets such as the HapMap [Int07] or the 1,000 Genomes Project [Pro10] and for these variants, haplotypes can be inferred using traditional techniques by leveraging the haplotypes from the reference dataset. We note that by using a reference dataset, we can predict haplotypes (or phase) at the common sites even for a single individual given the genotypes at the common sites for the individual.

Then the remaining variants (mostly rare variants) can be attached to the haplotypes by considering reads that span both the rare variant and a common allele for which the haplotypes have been inferred. The inference of the haplotypes can be performed using a modified dynamic programming algorithm that forces the haplotypes at the common variants to match the haplotype inferred from the genotypes.

We take the phased haplotype and treat them as a pair of very long reads with a gap at each site which is not present in the reference sample. We have two ways to place these "reads", namely assign one of the phased haplotype to one of the final haplotypes, say, h_0, the other phased haplotype to h_1, or the other way around. For each placement, we then apply the dynamic programming algorithm on the set of paired-end reads to infer the rare variants missed by the phased haplotypes. We need to use a modified dynamic programming algorithm where the phased haplotypes also need to be taken into consideration for the MEC since we initially have placed them. The placement with the minimum MEC will be the optimal placement and the corresponding reconstructed haplotypes are the

optimal haplotypes. Weights can also be applied to the dynamic programming algorithm when the MEC is computed. The weights can be determined according to our belief of the relative accuracies of the traditional techniques and the HTS technology respectively. Then when the MEC is summed over the paired-end reads and the phased haplotypes, different weights are assigned to the number of errors from the paired-end reads and the phased haplotypes accordingly.

We can estimate how effective this approach would be by considering how often any given variant is covered by a read which also covers an additional variant. We show such probabilities in Table 2.4 as well as the probabilities that the variants are covered by more than one read. As we can see, the probability increases as the coverage ratio or the standard deviation increases. With 40 times coverage and 500 base-pair standard deviation, the probability of a SNP being attached to other SNPs at least once is as high as 92%, and at least twice is also high as 83%. Therefore with even a moderate amount of coverage, most variants are covered by at least one read to another variant when the standard deviation of the insert length is big enough. Thus the combined strategy of using a traditional approach to infer haplotypes using the genotypes at the common variants combined with assembly of the rare variants using the sequence reads is a practical approach for inferring haplotypes.

2.2 Haplotype Phasing with Imputation using Sequencing Data

The haplotype assembly algorithms based on sequencing data worked well for sequencing studies where the sequencing coverage is high and the reads are long [LSN07], but perform very poorly for studies where the sequencing coverage is

	(10, 5)	(10, 50)	(10, 500)	(20, 5)	(20, 50)	(20, 500)	(30, 5)	(30, 50)	(30, 500)
≥ 1	41%	56%	65%	45%	66%	82%	46%	70%	89%
≥ 2	35%	38%	39%	41%	54%	62%	43%	62%	75%
≥ 3	29%	26%	23%	38%	44%	44%	41%	54%	61%
≥ 4	23%	19%	16%	35%	35%	32%	39%	46%	48%
	(40, 5)	(40, 50)	(40, 500)						
≥ 1	47%	73%	92%						
≥ 2	44%	66%	83%						
≥ 3	43%	60%	71%						
≥ 4	41%	54%	60%						

Table 2.4: The probability of a SNP attached to other SNPs more than once, twice, three times, four times, for different (coverage ratio, standard deviation) settings.

low or the reads are short. Our experiments in the previous section also reveals the deficiency of the haplotype assembly algorithms.

We propose a new method for haplotype inference from sequencing reads, Hap-seq, which combines information from a reference dataset with the information from the reads. We formulate the problem of haplotype inference using a likelihood framework. We use a hidden Markov model to represent the likelihood model of the predicted haplotypes given the reference datasets, which is similar to imputation methods [MHM07a, LWD10, KZE10], and we compute the posterior probability of the predicted haplotypes given the reads averaging over all possible assignments of the reads to chromosomal origin. Since consistency with the reference dataset and the read errors are independent, our joint likelihood is the product of these likelihoods. We present a dynamic programming algorithm for predicting the optimal haplotypes under this joint likelihood. This dynamic programming algorithm nests the dynamic programming algorithm over the reads with a dynamic programming algorithm over the reference haplotypes.

2.2.1 Likelihood Model

2.2.1.1 Overview

Previous methods for haplotype assembly [HCP10] focused on finding the haplotype and the assignment of reads to each haplotype such that the number of conflicts between the reads and the predicted haplotypes is minimized. In our framework, we formulate the same intuition by computing the likelihood of a pair of haplotypes as the probability of the reads given the haplotypes using a simple error model averaged over all possible partitions of the reads into chromosomal origin. Thus, we compute the likelihood with respect to the reads

$$\mathit{likelihood}_{\text{reads}} = P(\text{reads}|hap_1, hap_2)$$

where hap_1 and hap_2 are the (unknown) haplotypes of an individual.

We also take into account information from the reference dataset. We compute the probability of a pair of haplotypes where each haplotype is represented as a mosaic of reference haplotypes which is the standard HMM model for imputation. Thus, we compute the likelihood with respect to the reference dataset, or the *imputation likelihood*,

$$\mathit{likelihood}_{\text{imputation}} = P(hap_1, hap_2|\text{reference})$$

Figure 2.3 shows the two types of data, sequencing reads and reference dataset.

Since these two types of data are independent, we represent the likelihood of a haplotype given the reads and the reference dataset by the joint likelihood which is the product of the two likelihoods

$$L(hap_1, hap_2) = \mathit{likelihood}_{\text{reads}} \times \mathit{likelihood}_{\text{imputation}} \propto P(hap1, hap2|\text{reads}, \text{reference})$$

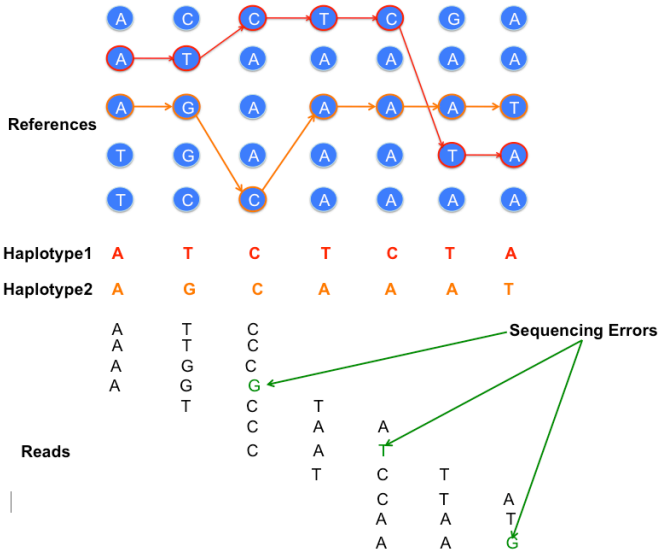

Figure 2.3: An illustration of reconstructing the pair of haplotypes as well as the imputation paths for each haplotype given a set of reference sequences and a set of sequencing reads which contains errors.

Our goal is to reconstruct a pair of haplotypes hap_1, hap_2 such that this likelihood is maximized. We call this objective function *MIR* (**M**ost likely **I**mputation based on **R**eads).

2.2.1.2 Likelihood with respect to Sequence Reads

We again followed the notation by [BB08a] for the haplotype assembly problem. Given a reference genome sequence and the set of reads containing sequence from both chromosomes, we align all the reads to the reference genome. However, unlike the haplotype assembly problem, where the homozygous sites (columns in the alignment with identical values) are discarded, we need to maintain the homozygous sites that are polymorphic in the reference as well as the heterozygous sites (columns in the alignment with different values). The sites are labelled as 0

or 1 arbitrarily.

A matrix X of size $m \times n$ can be built from the alignment, where m is the number of reads and n is the number of SNPs (defined as the total number of positions which are either polymorphic sites in the reference and/or heterozygous in the sample). The i-th read is described as a ternary string $X_i \in \{0, 1, -\}^n$, where '$-$' indicates a gap, namely that the allele is not covered by the fragment (again following the notation of [BB08a] for clarity). The *start position* and *end position* of a read are the first and last positions in the corresponding row that are not '$-$', respectively. Therefore the '$-$'s in the head and tail of each row will not be considered as part of the corresponding read. However, there can be '$-$'s inside each read which correspond to either missing data for single reads or gaps connecting a pair of single reads (called *paired end reads*). Reads without "$-$" are called *gapless reads*; otherwise they are called *gapped reads*.

The goal here is to describe our likelihood model of haplotypes with respect to reads. For this purpose, we should first describe the notion of *partial haplotype* which will be the unit of computation in our dynamic programming algorithm.

A partial haplotype is the prefix of full length haplotypes which ends with a suffix r of length k and the suffix starting position is i. For example, consider a full length haplotype "00010100010". When $i = 4$ and $k = 4$, the partial haplotype "0001010" has suffix $r =$"1010". Similarly, $r =$"0100" is the suffix of the partial haplotype "00010100" when $i = 5$ and $k = 4$. Since there are two chromosomes, we must consider two partial haplotypes and we refer to their two suffixes as r_1 and r_2. We note a difference from the haplotype assembly problem described in [HCP10] where r_1 and r_2 are always complementary because only heterozygous sites are considered. However, in our problem, we consider both homozygous sites and heterozygous sites and therefore r_1 and r_2 are not necessarily complementary.

Let $H(i, r_1)$ and $H(i, r_2)$ be the set of partial haplotypes which end with suffix r_1 and r_2 starting at position i, respectively. These sets contain 2^{i-1} haplotypes. We define $R(i, r_1, r_2)$ as the likelihood of the reads with starting position no greater than i that are generated from the pair of partial haplotypes $h^1 \in H(i, r_1)$ and $h^2 \in H(i, r_2)$ which maximizes the likelihood of the reads. The key idea behind our approach is that we will use dynamic programming to compute this quantity for larger and larger values of i eventually allowing us to identify complete haplotypes with the highest likelihood.

Since reads are independent, we can decompose our computation of $R(i, r_1, r_2)$ into the likelihood with respect to the reads starting at each position. Let $R'(i, r_1, r_2)$ be the likelihood only for the reads starting at position i. For every position, we assume the reads span at most k sites, thus all partial haplotypes with the same suffixes r_1, r_2 staring at position i ($h^1 \in H(i, r_1)$ and $h^2 \in H(i, r_2)$) will have the same value for $R'(i, r_1, r_2)$. Each read can originate from only one of the chromosomes corresponding to either r_1 or r_2. Since we do not know the origin of the reads, we consider all possible partitions of the reads. In each partition, a read is assigned to either r_1 or r_2 but not both. We can compute the number of mismatches between the reads and r_1 and r_2 and compute a likelihood given a partition using the following:

$$R'(l, i, r_1, r_2) = e^{E_l(i, r_1, r_2)}(1 - e)^{K(i) - E_l(i, r_1, r_2)}$$

where $R'(l, i, r_1, r_2)$ is the likelihood corresponding to the l-th partition and $E_l(i, r_1, r_2)$ is the number of mismatches for the l-th partition, $K(i)$ is the total count of the alleles of all reads starting at position i, e is the sequencing error rate. $R'(i, r_1, r_2)$ is sum of the likelihoods of all the partitions weighted by the probability of each partition, namely

$$R'(i, r_1, r_2) = \sum_{l=1}^{2^{a_i}} R'(l, i, r_1, r_2) P(\text{partition } l) = \sum_{l=1}^{2^{a_i}} R'(l, i, r_1, r_2)/2^{a_i} \qquad (2.2)$$

assuming there are totally a_i reads starting at position k, 2^{a_i} is the total number of possible partitions and the probability of each partition is equal to $\frac{1}{2^{a_i}}$.

Consider the two complete haplotypes $h^1_{\max} \in H(i, r_1)$ and $h^2_{\max} \in H(i, r_2)$ that maximize the likelihood of the reads and by definition, their likelihood is $R(i, r_1, r_2)$. For these haplotypes, R and R' have the following relationship:

$$R(i, r_1, r_2) = \prod_{f=1}^{i} R'(f, r'_1, r'_2) \qquad (2.3)$$

where r'_1 and r'_2 are length k suffixes of the partial haplotypes of $h^1_{\max}$ and $h^2_{\max}$ with suffix starting position f.

R' is computed for every possible suffix at each position of the dynamic programming. The dynamic programing uses the value of $R'(i, r_1, r_2)$ and the four values of $R(i-1, x, y)$ where the suffix of x is a prefix of r_1 and the suffix of y is a prefix of r_2 to compute $R(i, r_1, r_2)$ (see [HCP10] for details). For high sequencing coverage, we can compute just the likelihood of the most likely partition which approximates Equation (2.2).

2.2.1.3 Likelihood with respect to Reference Dataset

We utilize an HMM model which is the basis of the widely used imputation methods [MHM07a, LWD10, KZE10]. Given a binary string r, in the HMM, for each SNP at position i, we consider the HMM state with value $S_{i,j}$ corresponding to the j-th reference haplotype h_j, where $1 \leq j \leq m$. We define the transition probability from the state $S_{i,j}$ to the state $S_{i+1,y}$ as $t_{(i,j),(i+1,y)}$ where $1 \leq y \leq m$.

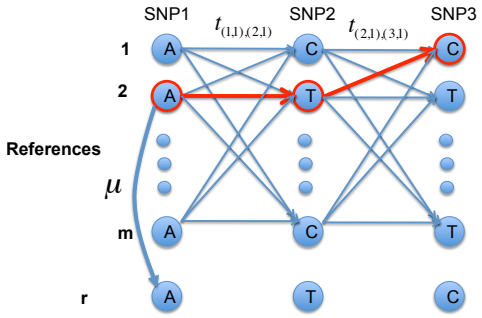

Figure 2.4: Example to illustrate the HMM (Hidden Markov Model) model. There are totally m reference sequences and 3 SNP locus. The given string r is "ATC" and the optimal imputation path is highlighted as red.

We assume that $t_{(i,j),(i+1,y)}$ is the same for all $y \neq j$. (Once the transition occurs, the transition probabilities to all possible states are equal.) We also define the emission probability from the state $S_{i,j}$ to the observed i-th SNP $r[i]$ as $\mu_{i,j}$. The emission probability models the mutations and we assume that the mutation rate for all the SNPs are the same. The emission probability $\mu_{i,j}$ is defined as the following:

$$\mu_{i,j} = \begin{cases} 1-\mu & h_j[i] = r[i] \\ \mu & \text{otherwise} \end{cases}$$

where μ is the mutation rate. We assume we know the mutation rate and the transition probabilities. An example of the HMM is shown in Figure 2.4.

Since we assume that we know μ as well as all $t_{(i,j),(i+1,y)}$'s, the most likely state sequence can be obtained by the Viterbi algorithm. We can also use the HMM to compute the likelihood of a partial haplotype. We define r as a length k binary string and $l(i, r, j)$ as the imputation likelihood of the partial haplotypes starting at position 0 and ending at position $i+k-1$, with suffix r, and whose imputation at position $i+k-1$ (the last position) is from the j-th reference sequence. $l(i, r, j)$

is the maximum value among many possible partial haplotypes having the same suffix r and the same ending HMM state $S_{i,j}$. Let $h_{l(i,r,j)}$ be the partial haplotype that maximized $l(i, r, j)$ which can usually be traced back in the algorithms.

2.2.2 Hap-seq

A naive algorithm to optimize MIR is to enumerate all possible pair of haplotypes and compute the likelihood, which requires time complexity $O(4^n)$, where n is the length of the haplotypes. Then for each pair, identify the number of mismatches between the reads and the haplotypes to compute the likelihood of the reads. After that, run the HMM to compute the likelihood of the imputation for both haplotypes. The complexity of this naive algorithm is obviously prohibitively large even for small n.

As the haplotype assembly problem can be optimally solved with a dynamic programming algorithm [HCP10] and the imputation problem can be optimally solved using an HMM, we next propose a hybrid method combining a dynamic programming algorithm and an HMM and we name the method **Hap-seq**. The basic idea is to use a dynamic programming algorithm operating on partial haplotypes with suffixes r_1, r_2, which are the prefixes of full length haplotypes similar to our algorithm for reconstructing haplotypes which maximizes the likelihood of the reads. However, for each suffix, we also introduce a state encoding the current reference haplotype. Similar to the approach above, at each position we compute the likelihood of reads starting at the position. We also use an HMM to compute the likelihood of imputation for the partial haplotypes using the information on the reference state. For each position i, we store the MIRs corresponding to different partial haplotype suffixes and imputation states in the dynamic programming matrix. Then we compute the MIR values for position

$i+1$ using the previous values, and repeat this process until we obtain the full length haplotypes. To compute the MIRs for position $i+1$ when extending the partial haplotypes, the MIR for the partial haplotypes from the previous position (i) is used in addition to the likelihood of reads for the new partial haplotypes (reads starting at position $i+1$) and an HMM is used to compute the imputation likelihood for the extended partial haplotypes. The new MIR for the partial haplotypes is then the product of the three values (likelihood from previous step, imputation likelihood, and likelihood with respect to reads). We next discuss our Hap-seq algorithm in more details.

We define $MIR(i, r_1, r_2, j_1, j_2)$ as the value of the MIR for the set of reads with starting positions no greater than i and the pair of partial haplotypes with suffix r_1, r_2, respectively, where the current references in the imputation model at position $i+k-1$ are the j_1-th and j_2-th reference sequences, respectively. We also define $MIR_{\max}(i, r_1, r_2)$ as the maximum MIR for $1 \leq j_1, j_2 \leq m$ given m haplotypes in the reference dataset.

We build a dynamic programming matrix and at each position i we store $MIR(i, r_1, r_2, j_1, j_2)$ for all r_1, r_2 and $1 \leq j_1, j_2 \leq m$, respectively. For each pair of r_1, r_2, we call the HMM to compute the likelihood of the imputation $l(i, r_1, j_1)$ and $l(i, r_2, j_2)$. MIR at position i can be computed as

$$\begin{aligned} MIR(i, r_1, r_2, j_1, j_2) &= R(i, r_1, r_2) \times l(i, r_1, j_1) \times l(i, r_2, j_2) \\ &\quad \text{where } h^1_{\max} = h_{l(i,r_1,j_1)} \text{ and } h^2_{\max} = h_{l(i,r_2,j_2)} \\ MIR_{\max}(i, r_1, r_2) &= \text{argmax}_{j_1,j_2} MIR(i, r_1, r_2, j_1, j_2) \text{ for } 1 \leq j_1, j_2 \leq m \end{aligned}$$

The best MIR is the maximum $MIR_{\max}(n-k, r_1, r_2)$ over all r_1, r_2 where n is the full length of the haplotypes.

The objective function contains terms $R(i, r_1, r_2)$, $l(i, r_1, j_1)$, and $l(i, r_2, j_2)$,

each of which is a maximum likelihood value maximized by finding the corresponding haplotypes ($h^1_{\max}$, $h^2_{\max}$), $h_{l(i,r_1,j_1)}$, and $h_{l(i,r_2,j_2)}$ respectively. It should be noted that in our definition, we constrain the problem using the condition $h^1_{\max} = h_{l(i,r_1,j_1)}$ and $h^2_{\max} = h_{l(i,r_2,j_2)}$, which forces the same haplotypes in each of the terms $R(i, r_1, r_2)$, $l(i, r_1, j_1)$, and $l(i, r_2, j_2)$. Thus we are searching for a single pair of haplotypes ($h^1_{\max} = h_{l(i,r_1,j_1)}, h^2_{\max} = h_{l(i,r_2,j_2)}$) which maximize the product of these three terms.

We initialize $MIR(0, r_1, r_2, j_1, j_2)$ by considering the reads starting at position 0. Given r_1, r_2, the HMM is called to compute the optimal likelihood $l(0, r_1, j_1), l(0, r_2, j_2)$ for the imputation of r_1, r_2, respectively. Then
$MIR(0, r_1, r_2, j_1, j_2) = R(0, r_1, r_2) \times l(0, r_1, j_1) \times l(0, r_2, j_2)$ for $1 \leq j_1, j_2 \leq m$.

2.2.2.1 Transition Probability

The partial haplotypes at position i can be obtained by extending the partial haplotypes at position $i-1$ with either a 0 or 1, as we consider homozygous sites and heterozygous sites. At position i, we again enumerate all length k binary strings for r_1 and r_2. Given a binary string r, $l(i, r, j)$ is obtained by calling an HMM on the j-th reference haplotype to find the likelihood for the imputation for the suffix r ending at position $i + k - 1$. In the HMM, since we need to consider both haplotypes at the same time, we build a state for each SNP which contains m^2 different values as $S_{(r_1,r_2),(i+k,j_1,j_2)}$ at position $i + k$ of the j_1-th and j_2-th reference haplotype h_{j_1}, h_{j_2}, where $1 \leq j_1, j_2 \leq m$ for all pairs of partial haplotypes with suffix r_1 and r_2, respectively. We define the transition probability from the state with value $S_{(r'_1,r'_2),(i+k-1,j_1,j_2)}$ to the state with value $S_{(r_1,r_2),(i+k,f_1,f_2)}$, where $1 \leq f_1, f_2, j_1, j_2 \leq m$ as $t_{(i+k-1,j_1,j_2),(i+k,f_1,f_2)}$. As the assignment of the two partial haplotypes to the reference haplotypes are independent, we obtain the

following formula:

$$t_{(i+k-1,j_1,j_2),(i+k,f_1,f_2)} = t_{(i+k-1,j_1),(i+k,f_1)} \times t_{(i+k-1,j_2),(i+k,f_2)}$$

where $t_{(i+k-1,j_1),(i+k,f_1)}$ and $t_{(i+k-1,j_2),(i+k,f_2)}$ are the standard transition probability used widely in the imputation methods. And we assume $t_{(i+k-1,j),(i+k,f)}$ is the same for all $1 \leq f, j \leq m, f \neq j$ for all possible r_1's and r_2's. $r_1' = (b, r_1[0, k-2])$ where b is 0 or 1. The brackets $[x, y]$ denote a substring starting from x and ending at y, both ends inclusive, where the index count starts from 0. A tuple of two strings denotes concatenation. Similarly, $r_2' = (b, r_2[0, k-2])$ where b is 0 or 1. For example, for k=4, r_1 = "0000", r_1' = "1000", we then have $r_1' = (1, r_1[0, 2])$.

2.2.2.2 Emission Probability

We also define the emission probability from the state with value $S_{(r_1,r_2),(i+k-1,j_1,j_2)}$ to the k-th observed SNPs in $r_1, r_2,\ r_1[k-1], r_2[k-1]$, as $\mu_{(r_1,r_2),(i+k-1,j_1,j_2)}$. Since the emissions from the two partial haplotypes are independent, we have the following formula:

$$\mu_{(r_1,r_2),(i+k-1,j_1,j_2)} = \mu_{r_1,(i+k-1,j_1)} \times \mu_{r_2,(i+k-1,j_2)}$$

The emission probability is an alternative of the mutation rate and we assume the mutation rate for all the SNPs are the same. The emission probability $\mu_{r,(i+k-1,j)}$ is defined as the following:

$$\mu_{r,(i+k-1,j)} = \begin{cases} 1-\mu & h_j[i+k-1] = r[k-1] \\ \mu & \text{otherwise} \end{cases}$$

where μ is the mutation rate which is the same for all the SNPs. Therefore if the two SNPs are identical, the emission probability is 1, otherwise it is μ. We assume we know the mutation rate and the transition probabilities.

2.2.2.3 Recursions

Given the transition and emission probabilities, the MIR at position i for partial haplotypes with suffix r_1, r_2 is computed as the following:

$$\begin{aligned} MIR(i, r_1, r_2, j_1, j_2) = \text{argmax}_{b_1,b_2,y_1,y_2} \Big(& \mu_{r_1,(i+k-1,y_1)} \times \mu_{r_2,(i+k-1,y_2)} \\ & \times t_{(i+k-2,y_1),(i+k-1,j_1)} \times t_{(i+k-2,y_2),(i+k-1,j_2)} \\ & \times MIR\big(i-1, (b_1, r_1[0, k-2]), (b_2, r_2[0, k-2]), y_1, y_2\big) \times R'(i, r_1, r_2) \Big) \\ & \text{for } \; b_1 \in \{0,1\}, b_2 \in \{0,1\}, 1 \leq y_1, y_2 \leq m, 1 \leq j_1, j_2 \leq m \end{aligned}$$

which allows us to compute $MIR_{\max}(i, r_1, r_2)$ for each i for every r_1, r_2.

2.2.2.4 Complexity

The complexity of the Hap-seq algorithm is $O(n \times m^4 \times 4^k)$ where 4^k is from the enumeration of all pairs of length-k binary strings and m^4 is from the HMM model on the m reference sequences, which is known to be reduced to m^2 by pre-computing and saving the transition probabilities [LWD10]. Therefore, the complexity of Hap-seq is $O(n \times m^2 \times 4^k)$ and typically m for imputation algorithms is around 100, for which Hap-seq is computationally feasible.

2.2.2.5 Read Length

At each position, we enumerate all possible binary strings of length k, where k is the maximum number of SNPs the reads contain. Since usually speaking the SNPs are far from each other, k is small. In our experiments, 99% of the reads contains no greater than 3 SNPs. Therefore we set a threshold as $k = 3$. For reads containing more than 3 SNPs, we simply split them. For example, for a read "0001000" starting at position 0, we split it as "000" starting at position 0, "100" starting at position 3 and "0" starting at position 6. We show later in our experiments with $k = 3$ Hap-seq can finish quickly.

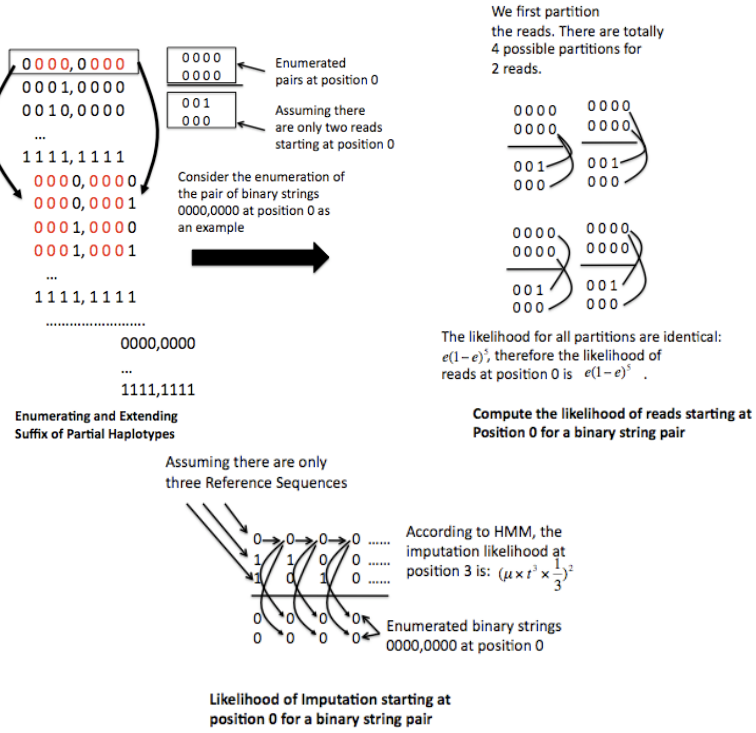

Figure 2.5: Example to illustrate the Hap-seq algorithm.

2.2.2.6 Illustrative Example

For illustration purpose, we show a running example of our Hap-seq algorithm in Figure 2.5. In Figure 2.5 (a), we show when $k = 4$, how we enumerate all length-4 binary strings at each position as suffixes of partial haplotypes and extend the suffix by one bit each time till the end of the haplotypes. As shown in the red color, the suffix of a binary string is identical to the prefix of the binary strings extended from it. In Figure 2.5 (b) and (c), we consider a pair of enumerated length-4 strings (0000, 0000) at position 0 as an example. We assume there are only two reads (001, 000) starting at position 0. In (b), we compute the likelihood of these two reads by considering all 4 possible partitions. The likelihood is computed according to Equation 2.2. In (c), we assume there are only three reference sequences. We apply HMM to compute the imputation likelihood for the pair of length-4 binary strings (0000,0000) at position 0. It is obvious that the best imputation path for both binary strings is through the first reference sequence without transition to other reference sequences. Then $MIR(0, 0000, 0000) = MIR(0, 0000, 0000, 1, 1)$, which is simply the product of the two likelihoods: $e(1-e)^5 \times (\mu \times t^3 \times \frac{1}{3})^2$.

Next at position 1 (we do not show this step in the above figure), assume we enumerate two binary strings as (0001, 0001), we can compute the likelihood $R'(1, 0001, 0001)$ of reads starting at position 1 by considering all possible partitions similar to the one shown in Figure 2.5. Since 0000 = (0, 0001[0,2]), 1000 = (1, 0001[0,2]), we can then compute the new MIR at position 1 as the following:

$$MIR(1, 0001, 0001, j_1, j_2) = \mu_{1,(4,y_1)} \times \mu_{1,(4,y_2)} \times t_{(3,y_1),(4,j_1)} \times t_{(3,y_2),(4,j_2)}$$
$$\times R'(1, 0001, 0001) \times \max \begin{cases} MIR(0, 0000, 0000, y_1, y_2) \\ MIR(0, 1000, 0000, y_1, y_2) \\ MIR(0, 0000, 1000, y_1, y_2) \\ MIR(0, 1000, 1000, y_1, y_2) \end{cases}$$

where $1 \leq j_1, j_2 \leq 3$ and $1 \leq y_1, y_2 \leq 3$. Then $MIR_{\max}(1, 0001, 0001) = \text{argmax}_{1 \leq j_1, j_2 \leq 3} MIR(1, 0001, 0001, j_1, j_2)$. We repeat the recursive procedure till we reach position $n - 4$ where n is the length of the full haplotypes.

2.2.3 Experimental Results

We perform simulation experiments to compare the performance of our Hap-seq algorithm and the standard HMM. The implementations of the two methods are as follows. The standard HMM is our own implementation of the IMPUTE v1.0 model which uses the pre-defined genetic map information for the transition probability. We use the genetic map data downloaded from the IMPUTE website. Since IMPUTE does not accept sequence read data, we allow our own implementation to accept sequence read by splitting the reads into SNP-wise information similarly to MACH [LWD10]. Our implementation runs the Viterbi algorithm and gives the most likely haplotype phasing based on the IMPUTE model. To make a fair comparison, for the part of Hap-seq that computes the imputation likelihood, we use the same IMPUTE v1.0 model using the same genetic map data.

We use 60 parental individuals of CEU population of HapMap Phase II data downloaded from the HapMap website. We perform a leave-one-out experiment to measure the accuracy of our method. We choose one target individual, generate

simulated sequence reads, and infer the genotypes and phasing of the target individual using the 59 individuals as the reference data set. We repeat this for all 60 individuals and the results are averaged. When we generate the data, we use the per-allele sequencing error rate of 1%. We assume that both IMPUTE and Hap-seq know the true error rate. In all datasets we generate, we assume a low coverage of 1x. That is, all heterozygous sites will be covered by two reads on average, since there are two chromosomes.

The first dataset we generate is the short region of 1,000 SNP sites at the beginning of chromosome 22. This region is approximately of length 1.8 Mb. We make an unrealistic assumption that each read covers a fixed number (F) of SNPs. Although this assumption is unrealistic, in this setting the performance gain of our Hap-seq over IMPUTE will be the most prominent because every read will contain phasing information when $F > 1$. We generate data for $F = 2$, $F = 3$ and $F = 4$. The results of switch error rate, which is the proportion of consecutive heterozygote genotypes that are incorrectly phased, for both algorithms are shown in the left of Table 2.5. As we can see, Hap-seq makes significant improvements over IMPUTE with respect to the number of switch errors. As F increases, the improvement becomes more significant. This is because as F increases, splitting reads loses more information that the SNPs in the same reads are from the same haplotype. Therefore Hap-seq has more advantages as the reads gets longer. One thing to note is that the accuracy of both methods drops at $F = 3$ compared to $F = 2$. This is because when F increases, the total number of reads decreases that can result in some regions not effectively covered by the reads. This raises an interesting question of what are the optimal read length and coverage that maximize the imputation and phasing accuracy given a cost, which requires further investigations.

F	IMPUTE	Hap-seq	Improvement
2	6.7%	6.2%	7.4%
3	5.4%	4.9%	8.2%
4	7%	5.7%	12.4%

W	IMPUTE	Hap-seq	improvement
100	5.36%	5.2%	2.8%
200	6.4%	6.25%	2.5%
500	7.2%	6.9%	3.5%
1000	7.8%	7.3%	6.5%

Table 2.5: Averaged switch error rate and the improvement of Hap-seq over IMPUTE.

The second dataset we generate is the same 1,000 SNP region where we generate a more realistic simulated reads that has a constant size (W) in bp. We vary W between 100, 200, 500, and 1,000 bp. Thus the number of SNPs contained in the reads is not fixed and can vary. However, as the SNPs are far from each other, over 99% of the reads contain no greater than three SNPs. The results of switch error rate for both algorithms are shown in the right of Table 2.5. We again observe over 2.5% improvement of Hap-seq over IMPUTE. The improvement is less significant compared with the last data set. This is because we do not fix the number of SNPs in the reads and thus there are many reads containing just one SNP. Hap-seq does not have any advantages over IMPUTE for these reads. We again observe a general increase in the improvement ratio when the read length increases.

Finally, we generate the paired-end read data of the whole chromosome 22, which contains 35,412 SNPs. We randomly select one individual and generate the paired-end sequence reads of size 1,000 bp in each end. The gap size is assumed to follow a normal distribution of mean 1,000 bp and the standard deviation of 100 bp. Therefore the number of SNPs contained in the reads can vary. We use 1X coverage and around 99% of the reads contain no greater than 3 SNPs, out of which 74% of the reads contain a single SNP. In order to parallelize our algorithm, we conduct a simple strategy. We split the chromosome 22 into chunks containing

1,000 SNPs each and run Hap-seq on each chunk in parallel only using reads in the chunk. In order to concatenate the reconstructed haplotypes from adjacent chunks seamlessly, we overlap adjacent chunks with a buffer region containing 200 SNPs. Therefore, the first chunk covers locus [0,1000], the second chunk covers locus [800, 2000], the third chunk covers locus [1800, 3000] and so on. Our intuition is that the haplotypes from adjacent chunks in the overlapped buffer region should be highly consistent with each other. Therefore, when we concatenate them, we select the best option which is able to minimize the differences between the haplotypes from adjacent chunks in the buffer region.

The program finished in 9 hours on a cluster processing the whole chromosome 22. We first compare the switch errors of Hap-seq and IMPUTE for each chunk containing 1,200 SNPs (the first and the last chunks contain different numbers of SNPs). As we can see in the left of Figure 2.6, it is obvious that for every chunk Hap-seq has lower switch error rate. For some chunks the improvement with respect to the number of switch errors is reduced by almost 80%. We also show the number of mismatches in the overlapping buffer regions for the haplotypes reconstructed by Hap-seq and IMPUTE in the right of Figure 2.6. We can see although there are certain regions with relatively high inconsistencies, most of the buffer regions are highly consistent with mismatches close to 0. We then concatenate the haplotypes from all the chunks. After the concatenation, we obtain a 2.79% overall error rate for Hap-seq on the whole chromosome 22. Compared to the 3.28% overall error rate for IMPUTE, this is a 15% improvement with respect to the number of switch errors. This again indicates that our Hap-seq algorithm is more effective in reconstructing the haplotypes using sequencing reads compared to IMPUTE.

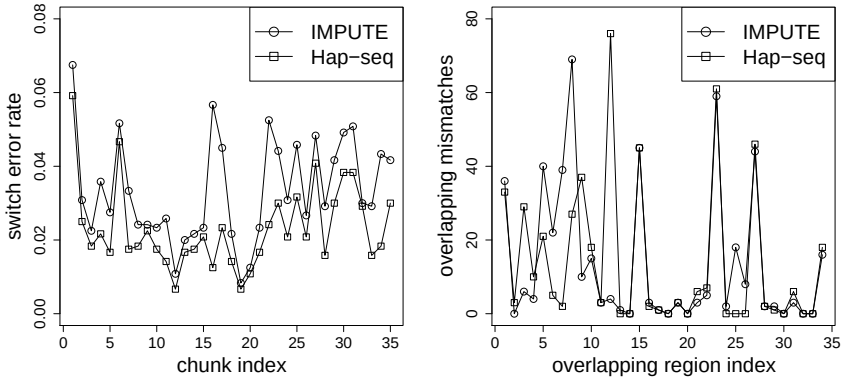

Figure 2.6: (Left) The switch error rate when using IMPUTE and Hap-seq for each chunk of length 1,200 SNPs for whole chromosome 22. (Right) The number of mismatches in the overlapping buffer regions for the haplotypes reconstructed by Hap-seq and IMPUTE.

CHAPTER 3

Efficient Algorithms for Detection and Reconstruction of Tandem Copy Number Variations

3.1 Background

Structural variations (SVs), such as insertions, deletions, and copy number variations (CNVs), have been shown to account for a large portion of genetic variance in both mouse and human genomes [IFR04][TSB05]. SVs are also known to contribute to phenotypic variation and have been implicated in a number of diseases [ZGH09]. Therefore these genetic variants can be utilized in a fashion similar to that of SNPs and may be useful when conducting association studies aimed at explaining mechanisms of complex diseases [MA07].

CNVs have been shown to make up around 12% of the human genome and various studies show their presence can affect gene expression, cause disease and alter the organism's phenotype [SFD07][ADV06][SLM07][GKB05]. For this reason, the problem of CNV detection has attracted a lot of recent attention. The efforts of many recent studies have been aimed at the detection of SVs and the prediction of their genomic regions [Sah09][IFR04] [VZC03][KUA07][KCD08][LCB08][TSB05]. However, the general problems of detecting and especially reconstructing CNVs still lack effective methods.

There have been several proposed methods for detecting CNVs based on comparative genomic hybridization (CGH) [LHA03][CLC08][DRO04][LJK05][MB08]. In CGH, both a genome of interest (donor genome) and a reference genome are hybridized to a tiling array. The genomes are labeled so that the intensity of each can be differentiated on the array. The ratio of intensities (donor/reference) at each spot provides an estimate of the gain or loss of the genomic sequence represented by the spot. This method, although successful in detecting CNVs, suffers from a number of limitations. The CGH approach lacks the ability to detect CNVs with high resolution. That is, the exact boundaries of the sequence which exists in variable copies are not distinguishable. Also, the CGH approach is unable to detect variations within the genomic copies [MA07] [CGJ09]. Nannya et al. [NSN05] and Wang et al. [WLH07] proposed more sensitive methods to improve resolution and genome coverage using whole-genome SNP genotyping arrays. But even these methods lack the ability to effectively reconstruct the regions of interest. In general, array-based methods may theoretically be able to predict the exact boundaries and number of copies of a particular region, but cannot be used to reconstruct the actual region as it appears in the donor genome.

The rising availability of next-generation sequencing (NGS) technologies offers an alternative way to detect CNVs. Next-generation sequencing provides a large number of short reads, as much as 40x coverage for a human individual. These reads can be mapped to a reference genome, in order to identify variations. A few recent studies have proposed methods to detect CNVs using datasets generated by NGS technologies [CGJ09][JRD10][MSB09][MFD10]. Simpson et al. [JRD10], using sequence data generated with inbred mouse strains, attempted to predict occurences of CNVs by using a Hidden Markov Model. Their method breaks the genome into a series of windows and determines the copy number state at each window. Adjacent windows that have the same copy number state are combined

to determine the full region of the CNV. Unfortunatly, the boundary resolution for this method is limited by the size of the window, which is typically at least 1 kilo-base. Chiang et al. [CGJ09] used a sliding window approach in order to identify genomic regions that are suspected to contain CNVs and to estimate the location of their boundaries. This method is able to predict the boundaries with greater resolution, because it is not limited by the choice of window size. Both of these methods have successfully identified true CNVs. However, their focus has been primarily on predicting the genomic sequence that exist in variable copies.

Medvedev et al. [MSB09] proposed a method to use discordant paired-end reads to identify structural variations. Discordant paired-end reads are reads mapped to the reference sequence in a way indicative of a structural variation. These discordant reads are clustered to provide high confidence for the occurrence of each structural variation. Medvedev et al. [MFD10] proposed an elegant method to detect copy number variations using paired-end reads. Similar to [MSB09], they first cluster discordant paired-end reads to identify CNV boundaries. Next they build a "donor graph", which represents the genome as a set of sequence blocks connected by a set of edges. The donor sequence can be reconstructed by walking along the edges of the donor graph. A maximal flow algorithm is applied to estimate the most-likely number of copies for each CNV. However, their method aims to solve the general CNV detection problem and is not specific for solving the CNV reconstruction problem.

We focus on the CNV detection and reconstruction problem for tandemly organized *de novo* CNVs. This type of CNV was shown to make up nearly 89% of all CNVs with size $\geq$ 10kb found in the mouse genomes [SCZ08]. Tandem CNVs have the properties that there are no gaps or very short gaps between the copies and there is only one copy in the reference CNV, while there are multiple

copies in the donor sequence. Therefore, according to the definition of Tandem CNVs, we are only trying to detect CNV gains but not CNV losses, since there can be only one copy of CNV in the reference sequence. This structure allows us to efficiently reconstruct the exact CNV copies. We call the CNV in the reference sequence *reference CNV* and the CNV in the donor sequence *donor CNV*. Each copy in the donor sequence can potentially have a different beginning and ending position (prefix and suffix), which can be considered as insertions and deletions. We detect CNVs by examining the mapping structure across the genome using discordant paired-end reads. A discordant read pair is one that maps to the reference in a way that suggests a CNV. These discordant read pairs serve as a signal for potential CNVs. We interrogate the regions in which we observe discordant pairs, in order to determine if a CNV is likely to have occured. Discordant pairs are clustered to obtain estimates of the CNV boundaries and read coverage is used to estimate the number of copies that exists in the donor. Unlike all the previous methods, our reconstruction algorithm utilizes unmapped reads in order to identify the exact boundaries of each of the predicted copies and subsequently reconstructs the CNV regions as they appear in the donor sequence. For each detected CNV, we estimate both the number of copies as well as the boundaries (breakpoints) of each copy in a donor genome. We then use these estimates in order to generate a reconstruction of the CNV region as it appears in the donor genome.

One limit of the method in [MFD10] is it only targets CNV detection in regions which are not repeat-rich, namely in regions whose sequences are relatively unique. This may greatly reduce the applicability of the method given the existence of many repeat-rich regions in the genomes. We address the problem by categorizing the occurrences of the CNVs into three categories:

- The reference CNV is in non-repeat region: In this case, all of the donor CNV junctions are mapped uniquely to the reference CNV, which can be uniquely identified and we don't have any false positive mapping positions.
- Middle area of the reference CNV is in the repeat region: In this case, we assume that the left most and the right most boundaries of the reference CNV are not in the repeat region and can be identified uniquely. However, repeats within the reference CNV lead to possible multiple mapping positions for the remaining donor CNV junctions, namely the two segments of a discordant paired-end reads can be mapped to multiple places in the reference CNV due to repeats.
- The whole reference CNV is in the repeat region: The left most boundary and the right most boundary of the reference CNV also have multiple mapping positions in the reference genome

Next we develop efficient algorithms for each of the three categories.

3.2 The reference CNV is in non-repeat region

3.2.1 CNV Detection

3.2.1.1 Discordant paired-end reads

Using next-generation sequencing, short reads are generated in pairs, where a short gap appears between the two reads and the distance of this gap is roughly known. The distance is called *insert length*. Discordant read pairs occur when the mapping of a paired-end read is not what is expected, given that there were no structural variations in the donor genome. An example to illustrate discordant

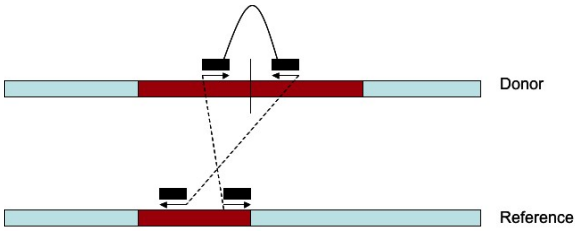

Figure 3.1: A discordant pair can imply the presence of a CNV.

read pairs is shown in Figure 3.1. In this figure, the reference genome contains one copy of the highlighted sequence, while the donor genome contains two tandem copies. A read is sampled from the donor such that the forward read is read from the end of the first copy and the reverse read is read from the beginning of the second copy. When mapped to the reference genome, these reads will map in an unexpected orientation. That is, the reverse read will map upstream of the forward read. By finding reads of this type, we can interogate their regions of mapping to find possible CNVs.

3.2.1.2 Clustering discordant pairs

After mapping a complete set of paired-end reads to a reference genome, we will find many discordant pairs. Given the high coverage, we expect that many discordant pairs will come from each CNV. Therefore, to avoid making duplicate CNV predictions and to bolster our confidence in each predicted CNV, we cluster discordant pairs so that each resulting cluster represents a potential CNV.

We define a simple greedy clustering procedure that clusters discordant pairs that explain the same CNV. We start with the complete set of discordant reads. We select one discordant read x from the complete set of discordant reads and find the set $\{y\}$ of all other discordant reads such that the difference in forward read mapping positions for x and y is no greater than M_I, which is the insert

length of the paired-end reads. We call the set of reads $x+\{y\}$ *explain* a potential CNV and remove them from the set of all discordant reads. Then we continue this processs until there are either no single discordant reads left in the set of all reads or there are no two single reads such that the difference in their forward read mapping positions is less than or equal to M_I. For each potential CNV, we now have a set of discordant reads and we estimate the boundaries by examining the set of mapping positions of these reads. We estimate the leftmost boundary b by taking the minimum of the reverse read mapping positions and the rightmost boundary $b + l$ by taking the maximum of the forward read mapping positions.

3.2.1.3 Estimating Copy-counts

Given a predicted CNV region c and the set of reads mapped to this region in the reference, we would like to determine how many copies d of the reference region are contained in the donor sequence. In order to determine d, we first define a function to calculate the likelihood of observing r reads within a region of length l, given a particular coverage level. For a region of length l, the probability of a read mapping to this region is $\frac{l}{G}$, where G is the length of the genome. We can calculate the probability of r reads mapping to the length l region by using the binomial distribution. Therefore, the expected number of reads mapped to a length l region is $\frac{Nl}{G}$, where N is the total number of reads generated from the whole genome. When N is very large, the binomial distribution can be very well approximated by the Poisson distribution. Therefore, we can use the density function for the Poisson distribution with $\lambda = \frac{N}{G}$, in order to calcluate the probability of r reads mapping to a length l region.

We expect that the length l region is represented d times in the donor sequence. However, we do not know d. We have only observed the number of reads

mapped to the length l region in the reference. Given this we define the following likelihood function.

$$\mathcal{L}(r, \lambda, l) = \frac{e^{-\lambda dl}(\lambda dl)^r}{r!} \tag{3.1}$$

By finding the d that maximizes the likelihood of r reads mapping to the reference region c, we determine the number of copies of c contained in the donor.

Medvedev et al. [MFD10] applied the same likelihood function defined in the Formula **??** and illustrated that it works well on Yoruban HapMap individual NA18507. Next we show this likelihood function may not always work. If we define $length(donor)$ as the total length of the donor CNV, we have $length(donor) = \sum_{i=1}^{d} l_i$ where d is the copy-counts, l_i is the length of the i-th copy and we define l as the length of the reference CNV. The likelihood function assumes all the copies in the donor sequence are of similar length, namely $l_i \approx l$ for all $1 \leq i \leq d$. Then $length(donor) \approx d \times l$. However, when the length of the copies differs from each other too much, namely the prefix and suffix of some i-th copies are big enough such that $l_i \ll l$, $length(donor)$ may decrease significantly from $d \times l$. Then we may have $||((d-k) \times l) - length(donor)|| < ||(d \times l) - length(donor)||$ for some $k \geq 1$, which implies that there is at least one copy of length less than l. Otherwise $(d \times l) - length(donor)$ should be 0. If the donor CNV is of length much less than $d \times l$, the true copy-counts would not maximize the likelihood function defined above and the copy-counts estimated will be inaccurate. We show later in our experiments that when the assumption that all the copies in the donor sequence are of similar length is violated, the likelihood function fails to identify the true copy-counts.

3.2.2 CNV Reconstruction

Once the CNV region is detected and the CNV copy-counts are estimated, we want to reconstruct the exact CNV copies as they appear in the donor sequence. We call the CNV in the reference sequence *reference CNV* and the CNV in the donor sequence *donor CNV*. The donor CNV can be denoted as $[p_1, s_1][p_2, s_2]...[p_n, s_n]$, where n is the copy-counts, $[p_i, s_i]$ indicates the i-th copy, p_i indicates the corresponding start position of the i-th copy in the reference sequence, s_i indicates the corresponding end position of the i-th copy in the reference sequence. We define $s_i|p_{i+1}$ as the junction between the i-th and $i+1$-th copy. The CNV Reconstruction problem thus can be stated as following: Given copy-counts n, reference CNV region $[b, b+l]$, where b is the starting position of the reference CNV and l is its length, identify all p_i and s_i for $1 \leq i \leq n$ and order them such that the number of mismatches is minimized when all the reads are mapped to the reconstructed sequence.

We use unmapped reads to detect the exact junctions of the copies. Given the high coverage ratio of the next generation sequencing data, each junction should be spanned by certain amount of reads. These reads won't map to the reference sequence and they indicate the start and end positions of the corresponding copies. We can split these unmapped reads and try to map both split parts to the reference sequence. If both parts map perfectly to the reference sequence (for simplicity, we assume all the reads are error-free), the mapped position indicates the corresponding end and start positions of the two adjacent copies. We show a simple example in Figure 3.2. As we can see, the reference CNV "CTGTCG" is copied three times in the donor sequence. The unmapped read TCGCT doesn't occur exactly in the reference sequence and it spans the junction between the first and the second CNV copy in the donor sequence. If we split the read into

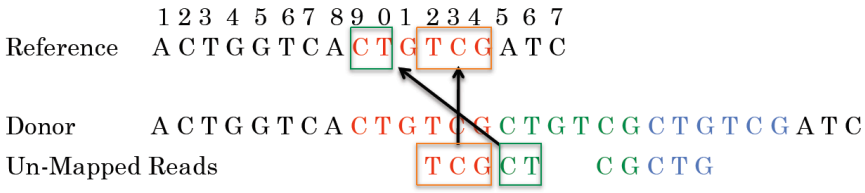

Figure 3.2: An example for reconstruction CNV. The reference CNV is "CTGTCG". The CNV is copied three times in the donor sequence.

two substrings TCG and CT, both of them map perfectly to the reference CNV. The end mapping position of TCG indicates the first copy ends at position 14 in the reference. The start mapping position of CT indicates the second copy starts at position 9. Next we show the detailed reconstruction process.

3.2.2.1 Unmapped Reads Identification

Since we are using unmapped reads to identify the boundaries of CNV copies, we first need to identify unmapped reads sampled from the CNV region. A naive method is to take all the unmapped reads, split them and map them to every identified CNV region. However, the sequencing technique may introduce a large number of unmapped reads due to sequencing errors. It is very inefficient if we take all of these reads and map them to every CNV region. The problem can be alleviated by using mapped reads. Although we don't know which donor CNV these unmapped reads come from, we can utilize the other parts in their corresponding paired-end reads to validate which donor CNV they are from. We first identify all the reads mapped to the reference CNV region, which can be done efficiently using any reads mapping tool such as MAQ [LRD08b]. Next we check the other parts of these reads and select all the unmapped ones, which are highly possible generated from the corresponding donor CNV region. The

number of such reads is much smaller than the total number of unmapped reads. We are then able to focus on only the unmapped reads generated from the correct donor CNV region.

3.2.2.2 Junction Validation

We split the unmapped reads and map them back to the reference sequence, as shown in Figure 3.2. A junction is valid only if it is validated by at least one unmapped reads. Therefore we can split an unmapped read at each internal position, which results in two split substrings r_1, r_2. r_1 is the suffix of the first copy and r_2 is the prefix of the second copy. We then map the two substrings back to the reference CNV. If both substrings mapped perfectly to the reference CNV (again, assuming the reads are error-free), the mapping positions in the reference CNV of r_1 and r_2 indicate the corresponding end and start positions of the two adjacent copies.

One problem for the validation is if the length of the split substring is short, it's highly possible that the split substring maps perfectly to the reference sequence by random chance and the substring may map to many positions. For example, in Figure 3.2, the unmapped read TCGCT is split to TCG and CT. The substring CT maps to both position 2 and 9. To address the problem, we define a *significant length threshold* t such that when we split the read, we only split the read at positions where the resulting two substrings are of length both no less than t. Alternatively speaking, we split the read at the positions in the range of [t, m-t] in the read, where m is the read length. Therefore a perfect map of a split substring to the reference sequence is highly unlikely occurring by chance. We call such a mapping a *significant mapping*. If a splitting results in two substrings which both have significant mappings, we call such a splitting a *significant*

splitting . The significant threshold can be determined by the following formula:

$$t = \arg\min_t \{\frac{L}{4^t} \leq \epsilon\} \tag{3.2}$$

where L is the length of the reference CNV region, ϵ is a small number, such as 0.05. The formula indicates that we use the minimum t such that the expected number of any length t substring occurring in the CNV region is no more than ϵ. For example, in Figure 3.2, L=17. If $\epsilon = 0.05$, according to Formula 3.2, we obtain $t \geq 4$. Therefore, the length of the substring as what we used in the example (length as 3 and 2 for the two split substrings) is too small to avoid possible mappings by random chance.

3.2.2.3 Donor Sequence Reconstruction

Given any order of the junctions, we are able to reconstruct a donor CNV. For example, if the order of the junctions is: $s_1|p_2, s_2|p_3, \ldots, s_{n-1}|p_n$, the reconstructed donor CNV is $[s, s_1], [p_2, s_2], \ldots, [p_{n-1}, s_{n-1}], [p_n, e]$, where s and e are the start and end positions of the CNV, respectively. However, not all the donor CNVs are valid. We need to follow a simple rule: for any two adjacent junctions $s_i|p_{i+1}$ and $s_{i+1}|p_{i+2}$, we need to have $p_{i+1} \leq s_{i+1}$, namely the starting position of a copy should be no greater than the ending position of the copy. For example, if we only have two adjacent junctions 100|200 and 150|50, the order of the two junctions can be only 150|50, 100|200. The corresponding reconstructed donor CNV is thus [s, 150][50, 100][200, e], where s and e are the start and end positions of the CNV, respectively. If we have order as 100|200, 150|50, the reconstructed donor CNV is [s, 100][200, 150][50, e], where the middle copy [200, 150] is invalid.

Notice there can be multiple orders satisfying this rule and therefore the donor

CNV may not be unique. For example, given junctions 100|200 and 300|50, s and e as the start and end positions of the CNV, we can have an order 100|200 and 300|50, where the reconstructed donor CNV is [s, 100][200, 300][50, e], or an order 300|50 and 100|200, where the reconstructed donor CNV is [s, 300][50, 100][200, e]. Both reconstructed donor CNVs are valid. We are not able to find the true donor CNV given the paired-end reads only. Therefore, we simply output all the possible donor CNVs.

3.2.3 Results

3.2.3.1 Simulation

In order to test our methods, we developed a simulation framework in which a donor genome is created by altering a given template genome by inserting non-overlapping CNVs. The number of copies for each CNV is selected uniformly at random from between 2 and 5 and the prefix and suffix of each copy is selected uniformly at random within a range that guarantees that the copy will be no smaller than 1000 base pairs. For each of the resulting donor genomes, paired-end reads are simulated. The reads are of length 36 bp and the inserts are chosen between 90 and 100 bp, uniformly at random. The full set of reads are mapped to the original reference genome using the MAQ mapper [LRD08b] and the resulting MAQ mapping files are then used by our detection and reconstruction methods.

We used this framework to generate one full mouse genome with 20 chromosomes, using build 36 of the mouse genome. For each chromosome we inserted 100 non-overlapping CNVs, avoiding regions with known CNVs, as described by [SCZ08]. Each generated genome contains 20 chromosomes each with 100 inserted CNVs, resulting in 2,000 CNVs in total. Each donor chromsome resulted in over 100 million reads. Table 3.1 summarizes the number of CNVs simulated within

CNV Length	Number	Copy-counts (2,3,4)
$l \geq 1,000,000$	80	31,26,33
$500,000 \leq l < 1,000,000$	390	120,139,131
$100,000 \leq l < 500,000$	975	319,335,331
$50,000 \leq l < 100,000$	217	70,79,68
$10,000 \leq l < 50,000$	247	78,89,80
$5,000 \leq l < 10,000$	47	15,14,18
$1,000 \leq l < 5,000$	44	12,15,17

Table 3.1: Summary of simulated CNVs. The number of CNVs belonging to each length class along with the number of copy-counts for each range are given.

each length range, as well as the number of these belonging to each copy-count category.

3.2.3.2 CNV Detection

We applied our detection algorithm to the set of 20 chromosomes containing 2,000 CNVs as described in section 3.2.3.1. When considering the fraction of detected CNVs in each chromosome, we find that our method is able to effectively detect on average 88.5% of the inserted CNVs, while accurately predicting the *copy-counts* for only 53% of these. This result shows that our method has relatively high power to detect the location of CNVs, but has low power to determine the correct copy-counts.

In order to understand why we have low power to predict copy-counts we compared the sets of CNVs for which we were able to predict the copy-counts and those for which we were not. For those which we are not able predict the copy-counts, we found that the number of reads mapping to the reference CNV

region was much smaller than expected given the length of the reference region and the true copy-counts. That is, our detection algorithm works under the assumption that the expected number of reads mapping to a reference region of length l for which the donor has n copies is lnc/R, where c is the coverage and R is the total length of a read pair. More simply, we expect that each of the ln base postions in the donor will be covered c times. Since each read pair covers R positions, we have lnc/R total reads expected. As the total length of the donor CNV region becomes much smaller than nl, our assumption is violated and we no longer have predictive power.

We examined the ratio of the total length of the CNV region in the donor to nl and found that for CNVs for which we could not detect the copy-counts this ratio was smaller when compared with the CNVs for which we could predict the copy-counts (mean of .82 vs. .86). Upon examining the distribution of these ratios, we find that their means are significantly different by t-test (p-value $\approx 10^{-15}$). As described in section 3.2.3.1, for each CNV the lengths of the copies were generated randomly. Given this, we can expect that as the number of copy-counts increases, the chance to have at least one copy that is significantly smaller than l increases. If this is the case, then we expect that on average our power to predict the true copy-counts will decrease as the number of copy-counts increases. This is what we observe and have summarized in table 3.2.

3.2.3.3 CNV Reconstruction

Once the predicted CNVs have been obtained, we attempt to reconstruct them as they appear in the donor sequence. We expect that reads spanning the junction between two adjacent copies will not map to the reference sequence. We utilize these reads in order to find the exact boundaries between copies. Given the

Copy-Counts	Detected	Predicted copy-counts
2	91%	78%
3	88%	49%
4	86%	33%

Table 3.2: Summary of the percentage of detected CNVs and predicted copy–counts broken down by true copy-counts. CNVs were considered to be detected when the begining and ending reference CNV positions were predicted within 10 base-pairs. In practice, the average deviation from the true positions was about 2 base-pairs. The percentage of predicted-copy counts is reported as the percentage of detected CNVs for which were were able to accurately predict the true copy-counts.

high coverage of the next generation sequencing data, each junction should be spanned by a at least one read. We can split these unmapped reads and try to map both parts to the reference sequence. If both parts map perfectly to the reference sequence (for simplicity, we assume all the reads are error-free), the mapped position indicates the corresponding start and end positions of the two adjacent copies.

We take the correctly identified CNV regions and their copy-counts from the previous step and then apply the reconstruction method to identify all junctions. To evaluate the accuracy of the reconstruction, we evaluate the accuracy of the identified junctions. We consider an identification as accurate if the identified position is within 100 bps of the true junction position. Given a junction $a|b$, we do not require that a successful identification always identifies both a and b correctly. Instead we allow partial identification, namely if we only identifie a successfully, we consider the identification of a as a success and the identification of b as a failure. Therefore for all junctions, we calculate the percentage of the

successful identifications. We summarize the performance of our method in Table 3.3 and 3.4.

As we can see in Table 3.3, the smaller the length of the CNV region, the higher the accuracy. This is because longer regions are more likely to contain long repeats. Thus even if we split the unmapped reads into two long enough parts, these parts may still randomly map to the repeat region. For CNV of length less than 50,000, the accuracy of our method is very good.

In Table 3.4, we observe that the accuracy increases as the copy counts decrease. We also show that the average length of the CNV regions for different copy-counts. As we can see, the lower the copy-counts is, the higher the accuracy is. Also the shorter the CNV region is, the higher the accuracy is. Copy-counts three has higher accuracy than copy-counts two since its CNV region is much shorter.

CNV Length	Accuracy
$l \geq$ 1,000,000	64.12%
500,000 $\leq l <$ 1,000,000	75.45%
100,000 $\leq l <$ 500,000	78.82%
50,000 $\leq l <$ 100,000	82.29%
$l <$ 50,000	94.17%

Table 3.3: CNV Length vs. CNV junction identification accuracy, where l is the length of the CNV in the reference.

Copy-Counts	Accuracy	Average Length
2	77.14%	519,571
3	79.44%	446,300
4	73.57%	525,717

Table 3.4: Copy Counts vs. CNV junction identification accuracy.

3.3 Middle area of the reference CNV is in the repeat region

In this case, we assume that the left most and the right most boundaries of the reference CNV are not in the repeat region and can be identified uniquely. However, repeats within the reference CNV lead to possible multiple mapping positions for the remaining donor CNV junctions, namely the two segments of a discordant paired-end reads can be mapped to multiple places in the reference CNV due to repeats. In order to reconstruct the CNV, we need to identify the correct mapping positions of the junctions. This can not be achieved alone by clustering discordant paired-end reads, since there will be multiple clusters at different positions for the same junction.

We solve the problem by estimating the length of the donor CNV first and then select one mapping position for each junction such that the reconstructed donor CNV has length closest to the estimated length. We estimate the length of the donor CNV using the number of reads that have mapping positions in the reference CNV. Since the left most and the right most boundaries of the reference CNV can be identified uniquely, we can find out the total number of reads having mapping positions in this region. As we showed before, the reads are generated with a Poisson distribution with respect to the length of the region l and the coverage ratio c. Given the total number of reads d having

mapping positions in the reference CNV region, we've shown $d \approx \frac{(fl+o)c}{length(read)}$ where f is the copy-counts, o is the sum of the number of mapping positions out of the reference CNV region for all length(read)-mer in the reference CNV region. Therefore, we can consequently estimate the length of the donor CNV region as $fl \approx \frac{d \times length(read)}{c} - o$. We can also estimate the copy-counts according to the process we described in the previous section using the formula $\frac{e^{-\lambda(fl+o)} \times (\lambda(fl+o))^d}{d!}$. The number of mapping positions needs to be selected are then determined by the copy-counts. The problem of selecting the correct mapping positions is formalized with the following:

Problem 1 *Given an estimate length l of the donor CNV, the position of the left most boundary b and the position of the right most boundary e, the copy-counts f, a set of n candidate junctions $J = [e_1|b_1, e_2|b_2, \ldots, e_n|b_n]$, select f-1 junctions $F = [e_{i_1}|b_{i_1}, e_{i_2}|b_{i_2}, \ldots, e_{i_{f-1}}|b_{i_{f-1}}]$ from the set J such that $||Length(F, b, e) - l||$ is minimized, where $Length(F, b, e) = (e_{i_1} - b) + (e_{i_2} - b_{i_1}) + \cdots + (e_{i_{f-1}} - b_{i_{f-2}}) + (e - b_{i_{f-1}}) = (e_{i_1} - b) + \sum_{j=2}^{f-1}(e_{i_j} - b_{i_{j-1}}) + (e - b_{i_{f-1}})$.*

A naive algorithm requires full enumeration of all the possible $f-1$ junctions from the set J, whose complexity is $O\binom{n}{f-1}$. Given n is usually as big as a few hundred to thousand, this complexity becomes too expensive to reconstruct a single CNV even for small f. Therefore, we design the following branch and bound algorithm to alleviate the expensive computation for solving the above problem.

We consider the junctions as nodes in a search tree. We randomly select a junction and then select one more junction at each step in a depth-first search manner. For any set of selected junctions, we derive an upper bound as well as a lower bound for *Length(F, b, e)* which we show are not dependent on the order

of the junctions. The two bounds are updated at each step of the search process. Based on these two bounds, a lower bound for the function $||Length(F, b, e) - l||$ is derived. A standard branch and bound algorithm is then applied to efficiently select the optimal set of junctions. Once a set of $f - 1$ junctions are selected, or we reach a leaf node of the search tree, the upper bound of search tree is updated as the value of $||Length(F, b, e) - l||$ which is then used to prune branches of the search tree. The upper bound is updated once a new leaf node is reached and a lower value of $||Length(F, b, e) - l||$ is obtained. The algorithm stops till there is no more branch to be explored. We next prove a set of lemmas used to design the branch and bound algorithm.

Given a set $F = [e_{i_1}|b_{i_1}, e_{i_2}|b_{i_2}, \ldots, e_{i_{f-1}}|b_{i_{f-1}}]$, an *order* of junctions in F is $[e_{j_1}|b_{j_1}, e_{j_2}|b_{j_2}, \ldots, e_{j_{f-1}}|b_{j_{f-1}}]$ such that there is an one-to-one mapping from j_h to i_t for $h, t \in \{1, \ldots, f-1\}$ and j_h, i_t can be different. We call j_h an *order index* for the corresponding junction. The order of the junctions indicates the order of the copies of donor CNV and there are totally $(f-1)!$ different orders. We first show an important property of the function $Length(F, b, e)$ with respect to the order of the junctions.

Lemma 1 *The order of junctions in F does not affect $Length(F, b, e)$.*

Proof: Since one order of junctions can be converted to any other different orders by continuously swapping the order index of two junctions each time, we just need to show this swapping operation does not change $Length(F, b, e)$. Assume we have one order $F = [\ldots, e_i|b_i, \ldots, e_j|b_j, \ldots]$ and another order $F' = [\ldots, e_j|b_j, \ldots, e_i|b_i, \ldots]$, where the only difference between F and F' is that the order index of $e_i|b_i$ and $e_j|b_j$ is swapped. Then assume F contains totally n junctions:

$$
\begin{aligned}
Length(F', b, e) &= (e_1 - b) \\
&\quad + \sum_{m=2}^{i-1}(e_m - b_{m-1}) + (e_j - b_{i-1}) + (e_{i+1} - b_j) \\
&\quad + \sum_{m=i+1}^{j-1}(e_m - b_{m-1}) + (e_i - b_{j-1}) + (e_{j+1} - b_i) \\
&\quad + \sum_{m=j+1}^{n}(e_m - b_{m-1}) + (e - b_n) \\
&= (e_1 - b) \\
&\quad + \sum_{m=2}^{i-1}(e_m - b_{m-1}) + (e_i - b_{i-1}) + (e_{i+1} - b_i) \\
&\quad + \sum_{m=i+1}^{j-1}(e_m - b_{m-1}) + (e_j - b_{j-1}) + (e_{j+1} - b_j) \\
&\quad + \sum_{m=j+1}^{n}(e_m - b_{m-1}) + (e - b_n) \\
&= (e_1 - b) + \sum_{m=2}^{n}(e_m - b_{m-1}) + (e - b_n) \\
&= Length(F, b, e)
\end{aligned}
$$

Therefore, the order of junctions in F does not affect $Length(F, b, e)$. ■

We next show that we can derive an upper bound as well as a lower bound for $Length(F, b, e)$ given any subset junctions of F.

Lemma 2 *The upper bound of* $Length(F, b, e)$ *is* $(e_1 - b) + \sum_{m=2}^{k}(e_m - b_{m-1}) + (e - b_k) + (n - k)(e - b)$ *given the size of* F *is* n*, a subset of selected junctions as* $F' = [e_1|b_1, e_2|b_2, \ldots, e_k|b_k]$*, where* $k \leq n$*.*

Proof: We proof this lemma by induction. When the subset is of size 1, assume $F' = [e_1|b_1]$, then we need to select $n - 1$ more junctions.

$$
\begin{aligned}
Length(F,b,e) &= (e_1 - b) + \sum_{m=2}^{n}(e_m - b_{m-1}) + (e - b_n) \\
&= (e_1 - b) + (e - b_1) + \sum_{m=2}^{n}(e_m - b_m) \\
&\leq (e_1 - b) + (e - b_1) + \sum_{m=2}^{n}(e - b) \\
&= (e_1 - b) + (e - b_1) + (n-1)(e - b)
\end{aligned}
$$

Similarly, when the subset is of size k, assume $F' = [e_1|b_1, e_2|b_2, \ldots, e_k|b_k]$, we have

$$
\begin{aligned}
Length(F,b,e) \leq (e_1 - b) &+ \sum_{m=2}^{k}(e_m - b_{m-1}) + (e - b_k) \\
&+ (n-k)(e-b) \qquad \blacksquare
\end{aligned}
$$

The same idea can be applied to derive the lower bound of $Length(F,b,e)$.

Lemma 3 *The lower bound of* $Length(F,b,e)$ *is* $(e_1 - b) + \sum_{m=2}^{k}(e_m - b_{m-1}) + (e - b_k)$ *given the size of* F *is* n*, and a subset of selected junctions as* $F' = [e_1|b_1, e_2|b_2, \ldots, e_k|b_k]$*, where* $k \leq n$*.*

Proof: We proof this lemma by induction. When the subset is of size 1, assume $F' = [e_1|b_1]$, then we need to select $n-1$ more junctions.

$$
\begin{aligned}
Length(F,b,e) &= (e_1 - b) + \sum_{m=2}^{n}(e_m - b_{m-1}) + (e - b_n) \\
&= (e_1 - b) + (e - b_1) + \sum_{m=2}^{n}(e_m - b_m) \\
&\geq (e_1 - b) + (e - b_1)
\end{aligned}
$$

Similarly, when the subset is of size k, assume $F' = [e_1|b_1, e_2|b_2, \ldots, e_k|b_k]$, we have

$$Length(F, b, e) \geq (e_1 - b) + \sum_{m=2}^{k}(e_m - b_{m-1}) + (e - b_k) \qquad \blacksquare$$

Given the upper bound up and the lower bound lo of $Length(F, b, e)$, we can compute a lower bound δ' for $\delta = ||Length(F, b, e) - l||$ as the following: (1) If $lo \leq l \leq up$, $\delta' = 0$. (2) If $l \leq lo$, $\delta' = lo - l$. (3) If $l \geq up$, $\delta' = l - up$. Therefore, we can apply the standard branch and bound algorithm to select the set of junctions which minimize δ. Our experiments show later that this branch and bound algorithm is much more efficient than the naive algorithm.

The above algorithm is able to select an optimal set of junctions to minimize the objective function $||Length(F, b, e) - l||$. However, this does not guarantee the selected junctions are the true junctions. This is because the estimated length is very close but usually not exactly the same as the true CNV length, since the estimated length is based on a limited amount of reads generated from the reference CNV region. Also the false positive positions of the donor junctions maybe selected by the above process, if some combination of them leads to the minimum difference between the estimated length and the reconstructed CNV length. Our experiments on simulated data show that on general the above process works pretty well even with these two limitations.

In Problem 1, $Length(F, b, e)$ is the length of the donor CNV based on our assumption $b_1 = b$ and $e_f = e$. Without this assumption, we can use discordant reads to identify the mapping positions for the left most junction containing b_1 and the right most junction containing e_f between the donor CNV and the genome. Then we can include these mapping positions in the candidate junction set J. All of the above lemmas and algorithms can be easily adapted to handle

this more general problem. Due to space limit, we do not show the generalization of the above algorithm.

3.3.1 Experimental Results

3.3.1.1 Experiments on Mouse Genome

We first illustrate the power of our method on real data. We used the data published by Sudbery et. al. 2008 [SSS08] where they sequenced the chromosome 17 of A/J mouse strain (95Mbp), using the Illumina technology [SSS08]. Chromosome 17 is biologically interesting for two main reasons. First, mouse major compatibility complex (MHC) resides on this chromosome [MHJ09]. Second, murine t-complex also resides in chromosome 17 which in some wild derived strains is responsible for transmission ration disorder [HJB09]. There are totally 56,019,759 paired-end reads sequenced and each segment of the paired-end read is 36bp long which makes the data has 22x coverage. The average insert length of the paired-end reads is around 120bp. Due to the fact we need all possible mapping locations (as indicated in previous studies [SB09][Sah09] [AKM09] [MSB09], [MFD10] for detecting CNV and SV, it is essential to use all the possible mapping positions) for each read, we map the reads using mrFAST [AKM09] [FCB10]. In the mapping we used the C57BL/6J (NCBI m37) as the reference to map the reads. The threshold of read mapping was set to $e \leq 2$ edit distance (allowing reads mapped with up to 2 mismatches and indels) in which reads will map to locations with sequence similarity higher than 94%(in all previous studies dealing with 36bp Illumina short reads, threshold of 2 mismatches is used). Lots of reads have multiple mapping positions and there are totally more than 8 billion mapping positions for these reads.

By using the discordant reads, we identify candidate regions for CNVs of

category two and three. For simplicity, we limit the length of the region within the range of [1kb, 10kb]. When we cluster the discordant reads, we require two adjacent reads in the same cluster overlap with each other. We also require the insert length for a paired-end read after being mapped to the reference sequence be greater than 500bp to be considered as discordant such that each copy is long enough. We only use clusters of reads with size greater than 7 to identify the candidate CNV regions. There 381 candidate regions for CNVs of category two, where the left most and the right most boundaries are determined by clusters of reads with unique mapping positions. We estimate the copy counts of these regions using the modified Poisson likelihood function and we identify a total of 5 regions with copy-counts no less than 2. One of these regions overlaps with the CNV regions reported by [SSS08], and 4 of them are new regions.

To further validate the 4 new regions, we apply our algorithm to reconstruct the CNVs. To our knowledge, there is no annotation of existing CNVs in repeat-rich regions, we are not able to evaluate the accuracy of the reconstruction directly from any known literature. Therefore, we validate our reconstruction using *unmapped* reads, which are reads that do not map to the reference sequence, mainly due to two reasons: (1) the reads contain sequencing errors, (2) the reads span the breakpoints of some structural variations. We say an unmapped read *supports* a breakpoint if it spans the breakpoint. To check if an unmapped read spans a breakpoint, similar to the work in [HFE10], we split the unmapped read into two substrings, and map both substrings to the two sides of the breakpoints. If both substrings match the corresponding side of the breakpoint, we say the unmapped read supports the breakpoint. However, the reconstructed breakpoints may be a little bit off from the true breakpoint due to the effects of reads mapping, clustering of the discordant reads, etc. Therefore we also check a 200bp neighborhood of the breakpoints. If the two split substrings of an unmapped read maps to the

	region 1	region 2	region 3	region 4	region 5
# unmapped reads	19.5	23	31	22	12
copy-counts	3	2	4	2	2

Figure 3.3: Averaged number of unmapped reads supporting the breakpoints in each of the 5 regions and their corresponding copy-counts.

neighborhood, we still consider the read supports the breakpoint. Then for each reconstructed breakpoint, we check the number of unmapped reads supporting it. Given the coverage as 22 times, the expected number of reads spanning any position in the genome, including breakpoints, is 22. In Figure 3.3, we show the averaged number of unmapped reads supporting the breakpoints in each of the 5 regions and their corresponding copy-counts. As we can see, for most of the regions, the number of unmapped reads supporting the breakpoints is very close to the expected number 22, indicating these regions and breakpoints are highly possible to be true.

3.3.1.2 Experiments on Simulated Sequences

As mentioned earlier, to our knowledge, there is no annotation of exact internal junctions of existing CNVs in repeat-rich regions, so to illustrate the performance of the algorithms we proposed, we also test our methods on simulated genome sequences. We conduct a set of experiments for each of the three cases with respect to different parameter. We first randomly generate a genome sequence which does not contain any long repeats. Then we insert CNVs at random positions in the simulated sequence. For case one, the reference CNV regions, prefixes and suffixes of the donor CNV copies as well as copy-counts are randomly generated. For case two, where the middle area of the reference CNV region contains repeats,

we first randomly select the reference CNV region. Next, we randomly sample pre-masked repeat sequences from the RepeatMasker website [Rep10] and insert these repeat sequences multiple times into the reference CNV region, avoiding the left most and right most boundaries. Finally, the prefixes and suffixes of the donor CNV copies are randomly generated. This process guarantees that the left most and right most boundaries of the reference CNV can be identified uniquely, while the internal junctions may fall within the repeat region. For case three, the simulation process is almost identical to the one for case two, with the only difference being that we insert repeats first and then randomly generate CNVs within repeat regions. For all experiments, we set the coverage ratio to 40 times, read length to 36bp and insert length within the range of 90bp to 100bp. We do not compare our method with any other existing CNV detection methods using high throughput sequencing technologies since none of them is able to handle CNVs in repeat-rich regions and also none of them is publicly available.

For the category two experiments, we construct repeat-rich regions by duplicating a repeat sequence, which is randomly sampled from RepeatMasker, 8 times, which is able to introduce enough complexes on the repeat structures. We set the copy-counts to 4 so that the number of mapping positions for each internal junction will be large. The above parameters are chosen just to make sure the problem is hard enough to show the advantage of our algorithm over the naive algorithm. Then we evaluate the performance of our branch and bound algorithm with respect to the length of the reference CNV region. The longer the reference CNV is, the more mapping positions the CNV is expected to have and the harder the reconstruction problem is. We vary the length of the reference CNV region as 10000bp, 5000bp, 2500bp. The experimental results are shown in Figure 3.4 and Figure 3.5. For each reference CNV length, we show the averaged results of our algorithm on ten experiments.

	10,000bp	5,000bp	2,500bp
Total Mapping Positions	1,307	267	77
BB Run Time (sec.)	19	1.48	0.07
BB searched orders	69,260	22,083	6,120
Full searched orders	371,259,885	3,136,805	73,150
BB/Full	0.019%	0.7%	8.4%

Figure 3.4: The efficiency evaluation of the branch and bound algorithm (BB) versus the full enumeration of all possible orders for reference CNV of different length. All the results are averaged on ten experiments.

In Figure 3.4, we illustrate the efficiency of our branch and bound algorithm. We first show the total number of mapping positions for the internal junctions. As we can see, as the size of the reference CNV increases, the number of mapping positions increases and the Problem 1 becomes more difficult. We also show the number of orders searched by our branch and bound algorithm and by the full enumeration and their ratio. The branch and bound algorithm has significant advantages over the full enumeration, especially for the length 10,000bp case, where our branch and bound algorithm only searches 0.019% of the orders of the full enumeration. Given this dramatic advantage, our algorithm has very short run times for all cases, while the run times of full enumeration are too long so we do not show them here.

In Figure 3.5, we show our algorithm is not only efficient, but also accurate. We compare the estimated internal junctions with the true internal junctions. If the difference is less than 100bp, which is the maximal insert length we use in our experiments, we consider the estimation as accurate. As we can see, with relatively short reference CNVs, our estimations are very accurate, especially for

	10,000bp	5,000bp	2,500bp
Accuracy	62.5%	94%	100%
Estimated Average Distance (Starting Position)	78	35	10
All-Position Average Distance (Starting Position)	6,179	3,033	1,569
Estimated Average Distance (Ending Position)	520	212	44
All-Position Average Distance (Ending Position)	5,489	2,788	1,318

Figure 3.5: The accuracy evaluation of the branch and bound algorithm for reference CNV of different length. All the results are averaged on ten experiments.

reference CNVs of size 2,500bp. This is because the number of mapping positions is small for short reference CNV regions, as shown in Figure 3.4, and for short reference CNV regions, the mapping positions tend to be close to each other. For cases where our algorithm makes incorrect estimations, we show that these estimations are still much better than random guesses. To illustrate this, we show the averaged distances between our estimated internal junctions and the true internal junctions, and the averaged distances between all possible mapping positions and the true internal junctions, which can be considered as the distance of random sampled positions. We measure the distance between the starting positions and the ending position separately and these distances are rounded into integers. As illustrated in Figure 3.5, the distances of our estimated junctions are only one percent of the distances of the random samples. Therefore, even though our algorithm may fail to estimate the true internal junctions, the estimations are still very close to the true junctions.

We also test our algorithm for other parameter settings, such as different copy-counts, different repeat sequence duplication times, etc., and we obtain similar results. Due to space limit, we do not show the results for other parameter settings.

3.4 The whole reference CNV is in the repeat region

When the left most boundary and the right most boundary of the reference CNV also have multiple mapping positions in the reference genome, the algorithm discussed in the previous section can not be applied directly. This is because we can not estimate the length of the donor CNV without knowing the true boundaries of the region. Therefore, we need to first identify the true boundaries of the reference CNV.

Unfortunately, we are not able to uniquely identify the true boundaries since they are mapped to multiple places in the reference genome. Therefore, the best we can do is to find a pair of mapping positions for the left most and right most boundaries and a copy-count value such that the reconstructed CNV has length closest to the estimated length, which is again $\frac{d \times length(read)}{c}$-$o$. Assuming the left most boundary is mapped to n positions in the reference genome, while the right most boundary is mapped to m positions in the reference genome, the problem can be formalized with the following:

Problem 2 *Given the set of mapping positions for the left most boundary* $B = \{b_1, b_2, \ldots, b_n\}$*, where* $b_1 < b_2 < \cdots < b_n$*, and the set of mapping positions for the right most boundary* $E = \{e_1, e_2, \ldots, e_m\}$*, where* $e_1 < e_2 < \cdots < e_m$*, select a pair* (b_i, e_j) *and a copy-counts* f *such that* $Dist(B, E, f, L, b_i, e_j) = ||(f-1) \times (e_j - b_i) + (e_m - b_1) - L||$ *is minimized for all* $1 \leq i \leq n$*,* $1 \leq j \leq m$ *and* $f \geq 2$*, where* L *is the estimated length of the donor CNV.*

We need to check all $O(n \times m)$ pairs to find the pair (b_i, e_j) minimizing $Dist(B, E, f, L, b_i, e_j)$. A naive method requires checking all $O(n \times m)$ pairs and testing many $u \geq 2$ for each pair until the best copy-counts is found. But in reality, most of these searches can be pruned.

We next design a more efficient algorithm. We try to select the length of the CNV, determined by the two boundaries, as well as the copy-counts at the same time. The selection process can be terminated a lot faster, based on our observation that to minimize $Dist(B, E, f, L, b_i, e_j)$, the copy-counts needs to remain the same or decrease when the length of the CNV increases. Therefore, we start our search from short CNV to long CNV, and decrease the copy-counts continuously until the copy-counts is less than 2, which is the stop criterion of the search process.

The algorithm works with the following lemma:

Lemma 4 *Given a pair* (b_i, e_j), $f = argmin_f Dist(B, E, f, L, b_i, e_j)$ *and a pair* (b'_i, e'_j), $f' = argmin_f Dist(B, E, f, L, b'_i, e'_j)$ *and* $e_j - b_i \leq e'_j - b'_i$, $Dist(B, E, f', L, b'_i, e'_j) \leq Dist(B, E, f, L, b_i, e_j)$, *we must have* $f' \leq f$.

Proof: We prove the lemma by contradiction. For the pair (b_i, e_j) and $f = argmin_f Dist(B, E, f, L, b_i, e_j)$, there are two cases:

1. f is the minimum copy-counts such that $(f-1) \times (e_j - b_i) + (e_m - b_1) \geq L$. Then $Dist(B, E, f, L, b_i, e_j) = (f-1) \times (e_j - b_i) + (e_m - b_1) - L$. Now assume $f' > f$, given $e_j - b_i \leq e'_j - b'_i$, we have $Dist(B, E, f', L, b'_i, e'_j) = (f'-1) \times (e'_j - b'_i) + (e_m - b_1) - L > (f-1) \times (e_j - b_i) + (e_m - b_1) - L = Dist(B, E, f, L, b_i, e_j)$, which is contradicted with $Dist(B, E, f, L, b'_i, e'_j) \leq Dist(B, E, f, L, b_i, e_j)$. Therefore, we can only have $f' \leq f$.

2. f is the maximal copy-counts such that $(f-1) \times (e_j - b_i) + (e_m - b_1) \leq L$. Then we must have $f \times (e_j - b_i) + (e_m - b_1) > L$. Since $f = argmin_f Dist(B, E, f, L, b_i, e_j)$, we must also have $Dist(B, E, f, L, b_i, e_j) \leq$

$Dist(B, E, f+1, L, b_i, e_j)$. Given $e_j - b_i \leq e'_j - b'_i$, we have $f \times (e'_j - b'_i) + (e_m - b_1) \geq f \times (e_j - b_i) + (e_m - b_1) \geq L$. Therefore $Dist(B, E, f+1, L, b'_i, e'_j) \geq Dist(B, E, f+1, L, b_i, e_j) \geq Dist(B, E, f, L, b_i, e_j)$. Now assume $f' > f$, namely $f' \geq f+1$, we have $Dist(B, E, f', L, b'_i, e'_j) \geq Dist(B, E, f+1, L, b'_i, e'_j) \geq Dist(B, E, f, L, b_i, e_j)$ which is contradicted with $Dist(B, E, f', L, b'_i, e'_j) \leq Dist(B, E, f, L, b_i, e_j)$. Therefore, we can only have $f' \leq f$. ■

The above lemma suggests that we can first identify a *minimum overlapping region*, which is the region that does not contain any sub-region. For example, for $b_1 < e_1 < b_2 < e_2$, the minimum overlapping regions are $[b_1, e_1]$ and $[b_2, e_2]$. $[b_1, e_2]$ is not a minimum overlapping region since it contains a sub-region $[b_1, e_1]$. Since the region is actually an one-dimensional interval, this can be done easily by finding the biggest b_i for each e_j such that $b_i < e_j$ and the minimum overlapping regions are all such (b_i, e_j)'s for each e_j. Then we can expand the regions in both directions to bigger regions. During the expansion process, the best copy-counts for each region can only remain the same or decrease. Once the copy-counts drop to below 2, we can stop the expansion. Since there are multiple minimum overlapping regions, we can start with the shortest one and we repeat the expansion process from each minimum overlapping region until we've checked all $O(n \times m)$ regions.

Once we find a pair (b_i, e_j) and a copy-count f such that $Dist(B, E, f, L, b_i, e_j)$ is minimized, we can apply the branch and bound algorithm discussed in the previous section to identify all the internal junctions.

As shown in Lemma 1, the order of junctions does not affect the length of the donor CNV. Therefore, for a given set of junctions minimizing $Length(F, b, e)$, we can not determine their true orders in the donor CNV since all these orders

are equally likely given the reads generated from the donor CNV. To reconstruct the donor CNV, we can simply select a random order of these junctions which leads to a corresponding reconstructed donor CNV.

3.4.1 Experimental Results

For the experiments on category three, we evaluate the performance of our pruning strategy with respect to the similar parameters as those for case two. In this experiment, we set the reference CNV length as 10000bp, the copy-counts as 4, and we vary the times we duplicate the repeat sequence sampled from RepeatMasker to create different levels of repeat-richness. We vary the repeat sequence duplication time as 1, 5, 10, 20, 30, respectively. The more times the sequence is duplicated, the more mapping positions the CNV is expected to have and thus the harder the problem is. In order to avoid noise, we only search for CNV regions above certain length threshold. This is because if we allow searching for short CNV regions, the short CNV regions tend to dominate the search since they are more likely to minimize the function defined in Problem 2. The most extreme case is when a CNV region is of length 1, the function will always be minimized by this CNV. Therefore, we use a relatively large threshold 8000bp. This indicates if we roughly know the length of the CNVs, our algorithm has better chance to identify them. Also we only insert one CNV for each repeat-rich region, which contains all duplications for the same repeat sequence. The experimental results are shown in Figure 3.6. All the results are averaged on ten experiments.

We compare the number of searches by the pruning algorithm and the non-pruning algorithm. As we can see, as the repeat-richness increases, the advantage of the pruning algorithm over the non-pruning algorithm becomes more significant. When we duplicate the repeat sequence 20 and 30 times, the pruning

	1	5	10	20	30
#Searches with pruning	23	2,302	2,620	10,054	14,779
#Searches without pruning	23	54,764	188,674	2,604,390	3,478,259
#Searches with pruning / without pruning	100%	4.2%	1.4%	0.39%	0.42%
Run time with pruning (sec.)	0	0	0	0	0
Run time without pruning (sec.)	0	1	2	48.5	83
Estimated ave. dist.	573	4,515	3,676	4,115	4,665
Random sampled ave. dist.	446	22,394	30,376	69,908	103,064
Estimated copy-counts	4	4	4	4	4
Randomly sampled copy-counts	1	1.03	1.3	1.6	1.7

Figure 3.6: The performance evaluation of the pruning algorithm for reference CNV of length 10000bp, copy-counts 4 and the repeat sequence duplication time as 1, 5, 10, 20, 30, respectively. All the results are averaged on ten experiments. The numbers are all rounded.

algorithm only makes less than 0.5% of the searches made by the non-pruning algorithm. As a consequence, the run time of the pruning algorithm is always less than 0.5 second while the run time of the non-pruning algorithm increases sharply as the repeat-richness increases. Again, to evaluate the accuracy of our method, we compare the distance between the estimated pair of boundaries to the true pair of boundaries. We compare this distance with the average distance of all pairs of positions satisfying the length threshold, which can be considered as the distance of randomly sampled pairs of boundaries. Our estimation is not very accurate, compared to the internal junction estimation for category two. But it is still a lot better than randomly sampled boundaries. Also our method is able to estimate the copy-counts always correctly while the copy-counts estimation from the randomly sampled pairs is almost always wrong.

CHAPTER 4

Genotyping common and rare variation using overlapping pool sequencing

4.1 Background

Recent advances in sequencing technologies have drastically reduced the cost of nucleotide sequencing [MEA05, HBB08] and are rapidly establishing themselves as very powerful tools for quantifying a growing list of cellular properties that include sequence variation, RNA expression levels, protein-DNA/RNA interaction sites, and chromatin methylation [WGS09, Sch08, MMM08, MWM08, JMM07, CFZ08]. Current sequencing machines typically split their throughput into a small number of independent lanes. If the sequencing capacity of a lane is larger than necessary for a sample, the simplest way to multiplex is to sequence a different sample in each lane. However, with the rapidly expanding throughput of the sequencing machines, it is often the case that for many applications the sequencing capacity per lane is much larger than what is necessary. This raises the need for multiplexing strategies that optimally use of the complete machine sequencing capacity to reduce the overall cost.

Two major multiplexing schemes have been proposed for high throughput sequencing. The first scheme is the barcoding approach in which each piece of DNA is labeled with a nucleotide sequence, the barcode, specific to each sam-

ple which is ligated in the process of creating the sequencing library. Together with the sample DNA, the sequencer also reads these barcodes and thus each read coming from the sequencer is labeled with the source of the sample. An alternative scheme to barcoding is the use of overlapping pools of samples for each run [ECG09, PP09, SAZ10]. In this scheme subsets of samples are mixed together into pools followed by a sequencing run within each pool. By combining the results of the sequencing with the information on which samples appeared in which pool, the mixed information from each pool can be "decoded" to obtain information on the sequence of each sample. Overlapping pool sequencing only makes sense when the coverage per lane is much higher than what is required per sample.

This is currently the case for sequencing sub-regions of the genome such as exome sequencing or targeted sequencing of a set of genes. Sequencing capacity per lane is constantly increasing and therefore it is plausible that multiplexing pools will benefit whole-genome sequencing in the future. We note that in Erlich et al [ECG09] the use of multiplexing has been proved in the lab, showing that this methodology is not merely a theoretical exercise. We note that barcoding and overlapping pools are orthogonal methods and can easily be combined to further increase multiplexing.

The current techniques for overlapping pool sequencing [ECG09, PP09, SAZ10] are based on group testing or compressed sensing schemes. While the exact details of the three schemes are different, each of them considers only discrete information from the pool. For example, these approaches examine a pool for the presence or absence of a specific sequence variant which combined with knowledge of the composition of the pools, can assign that variant to a specific sample. Because the information considered by each method is discrete, we refer to these approaches

as combinatorial approaches to overlapping pool sequencing. Their main limitation is that they are only applicable to detect rare variants. If a variant is common in the population, it will be present in almost every pool, causing the above pooling schemes to fail in identifying which subset of the samples contains the common variant.

We present an alternate scheme for sequencing using overlapping pools which, unlike all previous approaches, is able to quantify both rare and common variation. The key idea underlying our scheme is that we formulate the pooling problem within a likelihood framework that provides several advantages over previous methods. Our scheme is flexible and can be applied to a wide variety of applications. We demonstrate this by applying the scheme to two very different applications, each of which takes advantage of the likelihood framework within our approach and is difficult to solve using previously proposed combinatorial methods.

The first application we consider is obtaining highly accurate genotype information for a set of individuals. Currently, genotype microarrays are the most accurate method for measuring individual genetic variation at a base-pair level at variable locations across the genome (Single Nucleotide Polymorphisms: SNPs). A typical array will collect up to 900,000 or more genotype calls at common SNPs across the genome. Using imputation techniques and a reference dataset such as the HapMap[hap10] or the 1,000 Genomes project, we can make predictions for the remaining common variants in the genome. While error rates of genotyping are usually less than 0.5% errors at imputed variants range from around 5% in Europeans, and it could be as high as 10-15% for non-European populations [hap10, MH10, PAG10]. Imputation accuracy is particularly poor for rare SNPs and for SNPs in regions of low linkage disequilibrium. We intro-

duce here a scheme for obtaining highly accurate genotype information on both common and rare SNPs by combining genotyping microarrays, imputation and sequencing in pools of samples.

This application is possible because our likelihood framework allows us to integrate the information from the imputation into the procedure to help us "decode" the information obtained from each pool. Furthermore, our scheme allows us to utilize the variant frequency information obtained in each pool. Our results show that our algorithm is capable of calling rare SNPs with high accuracy, but in contrast to previous multiplexing methods, it can also call common SNPs with high accuracy, by combining the imputation data with the pooling scheme. In fact, the same experiment which can obtain genotype information for rare variants cambined with imputation can obtain genotype information at the common variants. Importantly, the outcome of our approach results in genotype information on the common variation which is more accurate than what is collected using microarrays. This application is particularly practical because much of the follow up sequencing of populations will be done in the same cohorts in which genome-wide association studies were performed. For these individuals, genotyping at common SNPs using microarrays has already been performed and for many of these studies only the regions of interest are targeted for sequencing, or exome sequencing is being performed, which makes multiplexing pools a practical approach at present.

The second application is to rapidly screen for fusion genes in cancer samples. Fusion genes play an important role in cancers and are caused by genomic rearrangements in a tumor that create new genes consisting of several exons from one gene followed by several exons from a second gene. Our application considers the sequencing of RNA obtained from cancer tumors with paired-end reads. The read

pairs of interest are ones that span exon boundaries with each read of the pair coming from a different exon. The majority of such read pairs will map to the same gene when aligned to the reference genome and. However, read pairs from fusion genes will map to two exons from different genes. One potential approach in identifying fusion genes is to search for read pairs that contain reads mapping to different genes. The main drawback of this approach is that it leads to a very high level of false positives making it difficult to distinguish actual fusions from experimental artifacts.

Our application will mix RNA from a large number of cancer tumors into overlapping pools and utilize the likelihood framework to decode which fusion genes come from which samples. In order to accomplish that, we extended the basic overlapping pool model to consider different levels of expression for each gene. This can be estimated from the data.

Our decoding scheme is based on a likelihood formulation which presents novel computational challenges compared to previous approaches. Each possible configuration (genotype assignment or gene-fusion assignment) is assigned a likelihood and the goal of the algorithm is to identify the most likely decoding. We identify good solutions for the problem by formulating a related problem as a linear program which we can efficiently solve.

4.2 Methods

We consider the scenario in which a set of N individuals are to be sequenced for any application such as a disease association study, or fusion-gene detection. The most straightforward approach would be to barcode the individual and sequence them separately. When N is large, or when the desired coverage is high, this ap-

proach is infeasible due to budget constraints. A few methods have been suggested to tackle this problem using a set of overlapping pools[ECG09, PP09, SAZ10]. These methods are based on the following generic idea. Let the sequences of the individuals be represented by a matrix $G = \{g_{ij}\}$ of dimension $N \times m$, where m is the length of the genome. $g_{ij} \in \{0, 1, 2\}$ is the number of occurrences of a genetic variant in position j of individual i - such variant could be a single nucleotide polymorphism (SNP), copy number variant (CNV), or a gene-fusion, as discussed in the introduction. The pooling based approach considers a $\{0, 1\}$ matrix A of dimension $T \times N$, representing a set of pools. Each row of A corresponds to a DNA pool; individual j participates in the $i - th$ row if and only if $A_{ij} = 1$. Thus, the matrix A provides a compact description of the study design. When the study is performed, under an error-free model, the pooling results are given as $Y = AG$. In principle, one can now decode the matrix by finding a solution to the set of equations $AX = Y$. In reality though the pooling results are not as accurate, and therefore current methods are using a rounded version of Y; for every i, j, we define $c_{ij} = 1$ if $y_{ij} > 0$ and $c_{ij} = 0$ otherwise. Thus, if we replace the SUM operation by an OR operation then $AG = C$. Using this information, Erlich et al.[ECG09],Prabhu et al.[PP09] and Shental et al.[SAZ10] provide a decoding algorithm which finds which individuals have $g_{ij} > 0$.

For the simplicity of the exposition, we will assume from now on that only one variant is considered, and so Y, C, and G are column vectors of length N.

A lower bound on decoding accuracy. Unfortunately, by collapsing the data to a $\{0, 1\}$ matrix resolution is lost, and therefore there is no hope in decoding all genetic variants from the pools if the number of pools is not large. Note that for a given variant, there are 3^N possible genotype vectors. The number of possible column vectors C_j is 2^T. Therefore, in order to be able to decode all

individual genotypes we need $2^T \geq 3^N$, or $T > N$. Even without rounding, the number of possible vectors B_j is at most $(2N)^T$, and therefore even in the error-free case we need $(2N)^T \geq 3^N$, or $T \geq \Omega(\frac{N}{\log N})$. In practice, the pooling decoding methods work well when the allele frequency is low, under an error-free model. For a variant of allele frequency α, the number of possible genotype vectors is $\binom{N}{\alpha N} \approx \left(\frac{1}{\alpha}\right)^{\alpha N}$, and therefore, we get that in the case where the rounded solutions are provided (the matrix C), we need $2^T \geq \left(\frac{1}{\alpha}\right)^{\alpha N}$, or $T \geq -N\alpha \log \alpha$, and if we are using the full information given by the matrix B, we need $T \geq -\frac{N\alpha \log \alpha}{\log N}$. Note that these are lower bounds, and it may theoretically be the case that a larger number of pools is required; however, it is easy to see that if A is chosen as a random matrix where each entry is 1 with probability 0.5, then the bounds given here are tight up to a constant factor (the proof is omitted from this version). Moreover, in [ECG09, PP09, SAZ10], it is shown that using the matrix C one can decode low allele frequencies ($\alpha = \frac{O(1)}{N}$) then $T = O(\log N)$ suffices, which is consistent with the bounds we provide here.

Since a random matrix provides a good decoding scheme in theory, we followed this intuition and generated a matrix A so that half of the entries in each row is 1 and the other half is 0. To obtain a better design matrix, we use local search; we repeatedly permute a random row and a random column and check to see if the Hamming distance between the permuted row/column and the other rows increased. If so, we keep the change, otherwise, we revert. After performing 1000 permutations, we result in a matrix whose rows and columns are farther apart, which improves our ability to decode.

Incorporating imputation into decoding As described above, from an information theoretical point of view, decoding the genotype vector is only possible when the allele frequency is low and therefore the genotype vector is sparse. For

this reason, both Erlich et al.[ECG09] and Shental et al.[SAZ10] make the connection between decoding and compressed sensing[Don10], where the requirement for the decoding success is based on the fact that the desired vector is sparse. We therefore suggest to incorporate imputation results into the decoding scheme; this allows us to overcome the information theoretical bound for the following reason. We can represent the true genotypes G as a sum of the (rounded) imputation predictions I, $i_{ij} \in \{0, 1, 2\}$, and a set of imputation residual errors R, $r_{ij} \in \{-2, 1, 0, 1, 2\}$, where $G = I + R$. Then, the observed data can be represented as the pools' results $Y = AG$, which is $Y = AI + AR$. Now, note that R is a sparse vector, and I is known; therefore, from a theoretical point of view, the above information theoretic lower bound does not hold on our case and there may be an algorithm that is able to decode the genotypes based on the sequencing and the imputation. In principle, we can solve for the imputation residual errors by solving the set of equations for $AX = Y - AI$. Once we obtain the residual vector, we can obtain the actual genotypes. In practice, as described below, we use the imputation dosage so our algorithm theoretically searches over the entire space, and not only over sparse vectors, but the search is pruned for vectors that are dense based on a likelihood model. As we show in the results section, this yields an improved imputation accuracy for high allele frequency SNPs.

4.2.1 Pooling using read counts

Our approach differs from previous approaches in that we are considering the matrix Y and not C. As discussed above, this should allow us a gain of approximately $\log N$ factor in the number of pools needed, at least for higher allele frequencies. However, in order to do so, we need to explicitly model the sequencing errors. The error model may be different, depending on the application at

hand. We will describe here the model we use for the detection of mutations (SNP calling). There are three main sources of noise that we include in the model:

1. There are slight differences in the concentration of each individual's DNA in each pool. This *pooling noise* is modeled as a normally distributed noise added to each non-zero element of A with mean 0 and variance σ_p. Thus, we set
$$\hat{A}_{ij} = A_{ij} + \mathcal{N}(0, \sigma_p) \; \forall \; A_{ij} \neq 0$$

2. There is a variance in the coverage of any specific region in the genome. We denote by L the length of the sequenced genomic region; if the total number of bases sequenced is λLN, then we expect that each base will be covered by λ reads on average. λ is often termed the expected *coverage*. We will denote the number of reads covering individual i by r_{i1}, r_{i2} (corresponding to the two chromosomal copies). Then, r_{ij} is Poisson distributed, with mean m_i. Prabhu et al.[PP09] showed that the m_i are approximately drawn from a Gamma distribution with $\alpha = 6.3$ and $\beta = \lambda/\alpha$ for Illumina Solexa sequencers. We note that for a given variant it is easy to infer the value of m_i, since it is shared across all individuals in all pools (assuming the pools are in the same lane). Thus, we have
$$m_i \sim \Gamma(\alpha, \beta), r_{ij} \sim Poisson(m_i)$$

3. The third source of error is sequencing error. The sequencing error rate depends on the location of the base in the read, but since the location of the base is uniformly distributed, we simply model the error rate by a constant probability ϵ for a substitution (1% is an acceptable estimate).

The above procedure produces a matrix $\hat{A}$ of noisy pools and a pair of vectors $\hat{R}^0, \hat{R}^1$ of noisy sequence reads; the number of sequence reads R_i^k is generated

by a Poisson distribution with an expectation that depends on the genotype g_i, and the coverage m_i, followed by a Binomial distribution to model the errors as explained above. R^0 corresponds to the reads with the major allele, while R^1 corresponds to the reads with the minor alleles. Note that even if $g_i = 0$, if $\epsilon > 0$, then expected number of reads with the minor allele will be greater than 0 because of errors. The pooling results are given by (Y^0, Y^1), where $Y^k = \hat{A}R^k$.

4.2.2 A likelihood model

Given the pooling results Y, we need to find a decoding algorithm that estimates G from Y. Our likelihood model takes into account both the error model, as well as population genetics data and external information when available. We decompose the likelihood $L(G; Pools)$ into several functions, and take their product as a composite likelihood.

Hardy-Weinberg Equilibrium. We first note that the overall allele frequency $\hat{p}$ of the SNP can be estimated as the average across all pools. We can now compute the Hardy-Weinberg (HW) probability of the observed genotypes $Pr^{HW}(G \mid \hat{p}) = 2^{n_1}\hat{p}^{n_2+n_1}(1-\hat{p})^{n_0}$, where n_0, n_1, n_2 are the genotype counts in G. Using Bayes law we have

$$Pr^{HW}(Pools \mid G) = Pr^{HW}(G \mid Pools)\frac{Pr(Pools)}{Pr(G)}$$

Assuming no prior information, we observe that maximizing $Pr^{HW}(Pools \mid G)$ is equivalent to maximizing $Pr^{HW}(G \mid Pools)$. We denote $f^{HW}(G) = Pr^{HW}(G \mid \hat{p}) = 2^{n_1}\hat{p}^{n_2+n_1}(1-\hat{p})^{n_0}$.

Likelihood of the observed reads. We compute the probability of the observed reads in the pools given G based on the noise model. Note that the only

unknown in the noise model is the concentrations of the individuals in the different pools. This is true since the coverage in any given region can be easily estimated as we use many pools in the same lane. Assume that λ is the coverage. Then, the number of reads with the minor allele (or major) contributed by individual j in pool i are Poisson distributed with $\hat{A}_{ij}\lambda G_j$ (or $\hat{A}_{ij}\lambda(2-G_j)$). Since the sum of Poisson distributions is Poisson distributed, we have that Y_j^k is Poisson distributed with a known expectation and thus we can write its likelihood. We denote this function by $f^{noise}(G)$. In order to find the concentration values $\hat{A_{ij}}$ we need to use external information. One such possibility could be to genotype small set of SNPs across the population and use those as the ground truth in order to tune the values of $\hat{A_{ij}}$. These SNPs provide a set of linear equations for the values $\hat{A_{ij}}$; for each pool we have one equation per SNP, and the number of variables is N. Therefore, genotyping as many as $O(N)$ SNPs and using a least squares approach guarantees an accurate estimate of $\hat{A_{ij}}$.

Likelihood of imputed data. Due to the bounds given on the possibility for detection, it is clear that without external information we will not be able to do much better than detecting rare SNPs. One natural choice for external data could be the genotypes of the individuals using microarrays. Today's genotyping technology is extremely cheap compared to sequencing, and the genotyping of thousands of individuals is feasible within a given study. The genotype information, however, provides the information about less than a million SNPs and another million CNVs across the genomes, while many other genetic variants are left unmeasured. To cope with this, imputation methods have been developed, in which nearby SNPs are used to impute unmeasured variants using the linkage disequilibrium structure of the genome[LA06, MHM07b]. However, this process is inevitably noisy, especially when imputing SNPs of low allele frequencies or

SNPs in regions of low linkage disequilibrium. Together with the pooling information we are able to provide a much more accurate calling of the imputed SNPs in all ranges of allele frequencies and linkage disequilibrium patterns. The output of the imputation method typically provides a distribution of the possible genotypes. For each individual j, we can assume that there is a given probability $h_i(0), h_i(1), h_i(2)$, where $h_i(j)$ is the probability that individual i has $G_i = j$. We can now use the imputation results for our likelihood model, by writing $f^{impute}(G) = Pr(G \mid imputation) = \prod_{i=1}^{N} h_i(G_i)$.

4.2.3 A Decoding Algorithm using Linear Programming

We use a linear program to bound the possible errors of each of the pools. If the coverage for the SNP is λ, we have that the pools should roughly satisfy $\lambda \hat{A} G = Y$. We can therefore solve the following linear program:

$$\begin{aligned} LP(G') = \min \quad & \sum_{i=1}^{T} x_i + \beta |G_i - I_i| \\ s.t. \quad & \lambda \sum_{j=1}^{N} \hat{a}_{ij} G_j - Y_i \leq x_i, \forall i \\ & \lambda \sum_{j=1}^{N} \hat{a}_{ij} G_j - Y_i \geq x_i, \forall i \\ & 0 \leq G_j \leq 2, j \in \{1, ..., N\} \end{aligned}$$

The linear program provides a lower bound on the best possible l_1 distance between $\lambda \hat{A} G$ and Y as well as returning a solution G which is close to the imputation prediction I. β is a parameter that trades off the relative importance of being close to the imputation vector compared to being consistent with the pools. Note that if Y_j is distributed as Poisson with expectation μ_j for which

$Y_j - \mu_j = x_j$, then

$$Pr(Y_j) = e^{\mu_j}\frac{\mu_j^{Y_j}}{Y_j!} = \frac{Y_j^{Y_j}e^{Y_j}}{Y_j!}e^{-x_j}(1-\frac{x_j}{Y_j})^{Y_j} \approx \frac{Y_j^{Y_j}e^{Y_j}}{Y_j!}e^{-2x_j}.$$

Therefore, we get that $f(v) \leq e^{-2LP(G')}\prod_{j=1}^{T}\frac{Y_j^{Y_j}e^{Y_j}}{Y_j!}$.

4.2.4 Application to Gene Fusion Detection

In order to detect gene fusions, we make several changes and extensions to the model presented above. The major additional complication in detection of fusion genes is that each sample may have a different expression level for a particular fusion gene. Even if we include the same amount of RNA from each tumor into each pool, the relative concentration of each gene will differ in each sample. However, this concentration is approximately constant across pools. Let e_{ij} be the normalized expression level of a particular variant j (in this case a fusion gene). Whether or not an individual i has the variant j is encoded as $G = \{g_{ij}\}$, $g_{ij} \in \{0, 1\}$. We define the matrix $H = \{e_{ij}g_{ij}\}$ as the concentrations of the samples and the results of the pools (assuming no noise) will then be $Y = AH$ instead of $Y = AG$ as in the genotyping application. In this application, we can also assume that the matrix G is sparse, but in order to perform the decoding, we must also estimate e_{ij} for the non-zero values of g_{ij}.

It is possible to estimate e_{ij} because they are constant across pools, however this introduces additional complexities in the optimization. We take advantage that fusion genes are very rare and most fusion genes are not shared across tumors. We constrain our optimization to allow for a maximum of k tumors to contain a given sample. We note that we only need to estimate the values of e_{ij} corresponding to non-zero elements of g_{ij}. To perform the optimization we enumerate over all $\binom{N}{k}$ possible genotype vectors and for each vector we estimate

the corresponding e_{ij} values.

Since optimizing the likelihood function for each possible genotype vector is computationally impractical, we solve a linear program as a method to quickly eliminate poor solutions. Let $A*$ be a matrix consisting of the only the columns of A corresponding to the non-zero entries of the genotype vector. If x is a vector which has a length the same as the number of non-zero elements in the genotype vector, the solution to $A*x = Y$ will be an approximate estimate of the values for e_{ij}. We can incorporate errors by adding a vector of all 1s to $A*$ and appending a term to x corresponding to the amount of errors expected in each pool. For the top 100 estimates obtained by using the pseudo-inverse, we then perform a grid search over the values of e_{ij} using the likelihood function described above.

4.3 Results and Discussion

4.3.1 Genotyping using Overlapping Pools and Imputation

We first report the results of applying our approach to genotyping individuals to obtain both common and rare variation using combining overlapping sequencing pools with genotyping and imputation. In this set of experiments, we utilize the 1958 Birth Cohort from the Wellcome Trust Case Control Consortium[Con07] data which contains approximately 1,500 individuals. These individuals were genotyped at approximately 500,000 SNPs. For every 10th SNP, we set the values of the genotypes to missing and applied MACH[LA06] using the HapMap data[Con05], an imputation algorithm, on these SNPs to make predictions. Since the SNPs were genotyped in the dataset, we can evaluate the accuracy of the imputation. We filter out any SNP with minor allele frequency lower than 5% since rare variants are easily genotyped using overlapping sequencing pools and

Parameter Values	Individuals = 100	
Num Pools	Imputation Accuracy	LP Accuracy
36	0.01	0.98
30	0.01	0.87

Table 4.1: Results of genotyping using overlapping sequence pools with imputation information.

the goal of these experiments is to evaluate the methods ability to genotype common variants.

We simulate applying our method by generating sequencing reads by generating reads consistent with the true values of the genotypes at the missing SNPs for each pool and then apply our method to make predictions of the genotypes incorporating the imputation information. We then measure the increase in accuracy of our prediction relative to the imputation information.

For our experiments we consider a total of 100 individuals mixed into 36 pools which is a reduction of the total number of sequencing lanes necessary by 1/3. We use a very high coverage of 150 per individual within a pool for our experiments under the assumption that the bottleneck is not the coverage, but the number of pools. We assume a sequencing error of 0.005. We measure the accuracy of the predictions by comparing the predicted genotypes to the true genotypes and only call a prediction correct if the genotypes are correct for all 100 individuals. We note that this is a very high standard and only 1 of our 100 SNPs have a correct imputation prediction. Our method has very high accuracy significantly improving over imputation. Table 4.1 summarizes the results.

4.3.2 Genotyping using Overlapping Pools without Imputation

We also apply our scheme to predict the genotypes without the imputation for rare variants such as those not found in the reference. The only difference in our methodology is that in the optimization problem, for the imputation vector we use a zero vector since we expect most individuals will not have the variant. This problem is actually much easier than the case of common alleles and we get perfect accuracy for the parameters above.

4.3.3 Cancer Fusion Gene Detection

We evaluate our approaches ability to detect cancer fusion genes using a similar simulation framework. In this application, RNA from different tumors is is mixed into overlapping pools and sequenced. In each pool we search for reads which cross exon boundaries from different genes and are evidence of fusion genes. Counts of these fusion genes in each pool are then decoded to identify the samples which contain the fusion genes.

We simulate this process by generating reads in a similar fashion to the genotyping without imputation simulations. We assume that we have 100 cancer samples where either 1, 2 or 3 of the samples contain a specific fusion gene. We assume a sequence error rate of 1% and vary the coverage and number of pools in our experiments. A difficulty in this application is that each individual has a different level of expression for each gene. We simulate this by randomly selecting an expression level in the range such that the concentration of the fusion gene in the RNA will differ by up to a factor of 10.

Table 4.2 shows the results of our cancer fusion gene detection simulation experiment. For each experiment, we report the fraction of the time that the

Parameter Values	# of Samples with Fusion		
(Num Pools, Coverage, Error Rate)	1	2	3
(10, 4 , 0.01)	0.980	0.760	0.340
(10, 12 , 0.01)	0.990	0.970	0.700
(10, 16, 0.01)	1.000	0.980	0.780
(10, 20, 0.01)	1.000	0.930	0.790
(10, 24, 0.01)	1.000	0.990	0.810
(10, 28, 0.01)	0.990	0.970	0.840
(4, 28, 0.01)	0.180	0.030	0.000
(6, 28, 0.01)	0.550	0.230	0.050
(8, 28, 0.01)	1.000	0.900	0.410

Table 4.2: Results of cancer fusion gene detection simulations. Each entry in the table is the fraction that the algorithm correctly identified the samples harboring the fusion gene.

algorithm identified correctly which samples contain the fusion gene.

CHAPTER 5

Conclusion and Future Work

5.1 Concluding Remarks

High-throughput Sequencing (HTS) technologies (also known as Next-Generation Sequencing) laid the foundations of many methods for human genetic variation related problems. We developed efficient algorithms for problems such as haplotype assembly, copy number variations detection and reconstruction, genotyping common and rare variations, based on the data generated by HTS. We show that our methods either improved the state-of-the-art solutions, or solved the problems that can not be addressed by the existing methods.

5.2 Future Work

Haplotypes are widely used to estimate the relatedness of human individuals in the form of IBD (Identity-by-Descent), where a pair of allele is IBD if they are identical copies of the same ancestral allele. There are many existing methods to estimate IBD between a pair of individuals. However, relatively less attention has been paid to the problem of estimating IBD among a group of individuals. We would like to develop new methods for this more challenging problem. Besides the algorithms, as IBD estimation mostly relies on genotype data, we would like to use HTS data and develop new models for the IBD estimation problem. The

estimated IBD can be used to construct pedigree for a set of individuals. However, the existing methods are relatively slow for even small number of generations. We would like to develop more efficient algorithms for pedigree reconstruction based on IBD estimation. What's more, we would like to develop an iterative procedure between IBD estimation and pedigree reconstruction such that we are able to improve the accuracy of both.

References

[ADV06] T.J. Aitman, R. Dong, T.J. Vyse, P.J. Norsworthy, M.D. Johnson, J. Smith, J. Mangion, C. Roberton-Lowe, A.J. Marshall, E. Petretto, et al. "Copy number polymorphism in Fcgr3 predisposes to glomerulonephritis in rats and humans." *Nature*, **439**(7078):851–855, 2006.

[AKM09] Can Alkan, Jeffrey M. Kidd, Tomas Marques-Bonet, Gozde Aksay, Francesca Antonacci, Fereydoun Hormozdiari, Jacob O. Kitzman, Carl Baker, Maika Malig, Onur Mutlu, S. Cenk Sahinalp, Richard A. Gibbs, and Evan E. Eichler. "Personalized copy number and segmental duplication maps using next-generation sequencing." 2009. Nature Genetics.

[BB08a] V. Bansal and V. Bafna. "HapCUT: an efficient and accurate algorithm for the haplotype assembly problem." *Bioinformatics*, **24**(16):i153, 2008.

[BB08b] B.L. Browning and S.R. Browning. "Haplotypic analysis of Wellcome Trust Case Control Consortium data." *Human genetics*, **123**(3):273–280, 2008.

[BHA08] V. Bansal, A.L. Halpern, N. Axelrod, and V. Bafna. "An MCMC algorithm for haplotype assembly from whole-genome sequence data." *Genome research*, **18**(8):1336, 2008.

[BHM09] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

[CFZ08] Shawn J. Cokus, Suhua Feng, Xiaoyu Zhang, Zugen Chen, Barry Merriman, Christian D. Haudenschild, Sriharsa Pradhan, Stanley F. Nelson, Matteo Pellegrini, and Steven E. Jacobsen. "Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning." *Nature*, **452**(7184):215–219, 03 2008.

[CGJ09] D.Y. Chiang, G. Getz, D.B. Jaffe, M.J.T. OKelly, X. Zhao, S.L. Carter, C. Russ, C. Nusbaum, M. Meyerson, and E.S. Lander. "High-resolution mapping of copy-number alterations with massively parallel sequencing." *Nature methods*, **6**(1):99, 2009.

[CIK05] R. Cilibrasi, L. van Iersel, S. Kelk, and J. Tromp. "On the complexity of several haplotyping problems." *Proceedings of the 5th Interna-*

tional Workshop on Algorithms in Bioinformatics. Lecture Notes in Computer Science, **3692**:128–139, 2005.

[CLC08] P.A. Chen, H.F. Liu, and K.M. Chao. "CNVDetector: locating copy number variations using array CGH data." *Bioinformatics*, **24**(23):2773, 2008.

[Con05] International HapMap Consortium. "A haplotype map of the human genome." *Nature*, **437**(7063):1299–320, 10 2005.

[Con07] Wellcome Trust Case Control Consortium. "Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls." *Nature*, **447**(7145):661–78, 6 2007.

[CZH08] Arthur Choi, Noah Zaitlen, Buhm Han, Knot Pipatsrisawat, Adnan Darwiche, and Eleazar Eskin. "Efficient Genome Wide Tagging by Reduction to SAT." *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics. Lecture Notes in Computer Science*, **5251**:135–147, 2008.

[Don10] D.L. Donoho. "Compressed Sensing." *IEEE Transactions on Information Theory*, **52**(4):1289–1306, 2010.

[DRO04] R.S. Daruwala, A. Rudra, H. Ostrer, R. Lucito, M. Wigler, and B. Mishra. "A versatile statistical analysis algorithm to detect genome copy number variation." *Proceedings of the National Academy of Sciences of the United States of America*, **101**(46):16292, 2004.

[ECG09] Yaniv Erlich, Kenneth Chang, Assaf Gordon, Roy Ronen, Oron Navon, Michelle Rooks, and Gregory J. Hannon. "DNA Sudoku–harnessing high-throughput sequencing for multiplexed specimen analysis." *Genome Res*, **19**(7):1243–53, 7 2009.

[FCB10] FarazHach and Fereydoun Hormozdiari, Farhad Hormozdiari Can Alkan, Inanc Birol, Evan E. Eichler, and S. Cenk Sahinalp. "mrsFAST a cache-oblivious algorithm for short-read mapping." 2010. Nature Methods.

[Fit77] W.M. Fitch. "On the problem of discovering the most parsimonious tree." *American Naturalist*, **111**(978):223–257, 1977.

[GKB05] E. Gonzalez, H. Kulkarni, H. Bolivar, A. Mangano, R. Sanchez, G. Catano, R.J. Nibbs, B.I. Freedman, M.P. Quinones, M.J.

Bamshad, et al. "The influence of CCL3L1 gene-containing segmental duplications on HIV-1/AIDS susceptibility." *Science*, **307**(5714):1434, 2005.

[Gus03] D. Gusfield. "Haplotype inference by pure parsimony." In *Proceedings of the Combinatorial Pattern Matching Conference*, pp. 144–155, 2003.

[hap10] "Integrating common and rare genetic variation in diverse human populations." *Nature*, **467**(7311):52–58, 09 2010.

[HBB08] Timothy D. Harris, Phillip R. Buzby, Hazen Babcock, Eric Beer, Jayson Bowers, Ido Braslavsky, Marie Causey, Jennifer Colonell, James Dimeo, J. William Efcavitch, Eldar Giladi, Jaime Gill, John Healy, Mirna Jarosz, Dan Lapen, Keith Moulton, Stephen R. Quake, Kathleen Steinmann, Edward Thayer, Anastasia Tyurina, Rebecca Ward, Howard Weiss, and Zheng Xie. "Single-molecule DNA sequencing of a viral genome." *Science*, **320**(5872):106–9, 4 2008.

[HCP10] D. He, A. Choi, K. Pipatsrisawat, A. Darwiche, and E. Eskin. "Optimal algorithms for haplotype assembly from whole-genome sequence data." *Bioinformatics*, **26**(12):i183, 2010.

[HE04] Eran Halperin and Eleazar Eskin. "Haplotype reconstruction from genotype data using Imperfect Phylogeny." *Bioinformatics*, **20**(12):1842–9, 8 2004.

[HFE10] D. He, N. Furlotte, and E. Eskin. "Detection and Reconstruction of tandemly organized de novo Copy Number Variations." *BMC Bioinformatics*, 2010.

[HHE12] D. He, B. Han, and E. Eskin. "Optimal Algorithm for Haplotype Phasing with Imputation using Sequencing Data." *Proceedings of the 16th Annual International Conference on Research in Computational Molecular Biology (Recomb 2012)*, 2012.

[HHF11] D. He, F. Hormozdiari, N. Furlotte, and E. Eskin. "Efficient algorithms for tandem copy number variation reconstruction in repeat-rich regions." *Bioinformatics*, **27**(11):1513–1520, 2011.

[HJB09] Bauer H, Willert J, Koschorz B, and Herrmann BG. "The t complex-encoded GTPase-activating protein Tagap1 acts as a transmission ratio distorter in mice." 2009. Nature Genetics.

[HZP11] D. He, N. Zaitlen, B. Pasaniuc, E. Eskin, and E. Halperin. "Genotyping common and rare variation using overlapping pool sequencing." *BMC Bioinformatics*, **12**, 2011.

[IFR04] A.J. Iafrate, L. Feuk, M.N. Rivera, M.L. Listewnik, P.K. Donahoe, Y. Qi, S.W. Scherer, and C. Lee. "Detection of large-scale variation in the human genome." *Nature genetics*, **36**(9):949–951, 2004.

[Int07] International HapMap Consortium. "A second generation human haplotype map of over 3.1 million SNPs." *Nature*, **449**(7164):851–61, 10 2007.

[JMM07] David S. Johnson, Ali Mortazavi, Richard M. Myers, and Barbara Wold. "Genome-Wide Mapping of in Vivo Protein-DNA Interactions." *Science*, p. 1141319, 2007.

[JRD10] Simpson JT, McIntyre RE, Adams DJ, and Durbin R. "Copy number variant detection in inbred strains from short read sequence data." *Bioinformatics*, **26**(4):565–567, 2010.

[KCD08] J.M. Kidd, G.M. Cooper, W.F. Donahue, H.S. Hayden, N. Sampas, T. Graves, N. Hansen, B. Teague, C. Alkan, F. Antonacci, et al. "Mapping and sequencing of structural variation from eight human genomes." *Nature*, **453**(7191):56–64, 2008.

[KUA07] J.O. Korbel, A.E. Urban, J.P. Affourtit, B. Godwin, F. Grubert, J.F. Simons, P.M. Kim, D. Palejev, N.J. Carriero, L. Du, et al. "Paired-end mapping reveals extensive structural variation in the human genome." *Science*, **318**(5849):420, 2007.

[KZE10] H.M. Kang, N.A. Zaitlen, and E. Eskin. "EMINIM: An Adaptive and Memory-Efficient Algorithm for Genotype Imputation." *Journal of Computational Biology*, **17**(3):547–560, 2010.

[LA06] Y Li and GR Abecasis. "Mach 1.0: Rapid Haplotype Reconstruction and Missing Genotype Inference." *Am J Hum Genet*, **S79**(2290), 2006.

[LBI01] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. "SNPs problems, complexity, and algorithms." *Proceedings of the 9th Annual European Symposium on Algorithms. Lecture Notes in Computer Science*, pp. 182–193, 2001.

[LCB08] S. Lee, E. Cheran, and M. Brudno. "A robust framework for detecting structural variations in a genome." *Bioinformatics*, **24**(13):i59, 2008.

[LHA03] R. Lucito, J. Healy, J. Alexander, A. Reiner, D. Esposito, M. Chi, L. Rodgers, A. Brady, J. Sebat, J. Troge, et al. "Representational oligonucleotide microarray analysis: a high-resolution method to detect genome copy number variation." *Genome Research*, **13**(10):2291, 2003.

[LJK05] W.R. Lai, M.D. Johnson, R. Kucherlapati, and P.J. Park. "Comparative analysis of algorithms for identifying amplifications and deletions in array CGH data." *Bioinformatics*, **21**(19):3763, 2005.

[LM09] C. Li and F. Manyá. "MaxSAT, hard and soft constraints." *Handbook of Satisfiability*, **185**:613–631, 2009.

[LRD08a] H. Li, J. Ruan, and R. Durbin. "Mapping short DNA sequencing reads and calling variants using mapping quality scores." *Genome research*, **18**(11):1851, 2008.

[LRD08b] H. Li, J. Ruan, and R. Durbin. "Mapping short DNA sequencing reads and calling variants using mapping quality scores." *Genome research*, **18**(11):1851, 2008.

[LSL02] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem." *Briefings in Bioinformatics*, **3**(1):23, 2002.

[LSN07] S. Levy, G. Sutton, P.C. Ng, L. Feuk, A.L. Halpern, B.P. Walenz, N. Axelrod, J. Huang, E.F. Kirkness, G. Denisov, et al. "The diploid genome sequence of an individual human." *PLoS Biol*, **5**(10):e254, 2007.

[LWD10] Yun Li, Cristen J Willer, Jun Ding, Paul Scheet, and Goncalo R Abecasis. "MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes." *Genet Epidemiol*, **34**(8):816–34, Dec 2010.

[MA07] Steven A. McCarroll and David Alshuler. "Copy-number variation and association studies of human disease." *Nature Supplement*, **29**:S37–S42, 2007.

[MB08] A. Mah and D. Brutlag. "New Genomics Technology: Copy Number Variation Analysis Methods." 2008.

[MEA05] Marcel Margulies, Michael Egholm, William E. Altman, Said Attiya, Joel S. Bader, Lisa A. Bemben, Jan Berka, Michael S. Braverman,

Yi-Ju J. Chen, Zhoutao Chen, Scott B. Dewell, Lei Du, Joseph M. Fierro, Xavier V. Gomes, Brian C. Godwin, Wen He, Scott Helgesen, Chun Heen Ho, Chun He Ho, Gerard P. Irzyk, Szilveszter C. Jando, Maria L. I. Alenquer, Thomas P. Jarvie, Kshama B. Jirage, Jong-Bum B. Kim, James R. Knight, Janna R. Lanza, John H. Leamon, Steven M. Lefkowitz, Ming Lei, Jing Li, Kenton L. Lohman, Hong Lu, Vinod B. Makhijani, Keith E. McDade, Michael P. McKenna, Eugene W. Myers, Elizabeth Nickerson, John R. Nobile, Ramona Plant, Bernard P. Puc, Michael T. Ronan, George T. Roth, Gary J. Sarkis, Jan Fredrik Simons, John W. Simpson, Maithreyan Srinivasan, Karrie R. Tartaro, Alexander Tomasz, Kari A. Vogt, Greg A. Volkmer, Shally H. Wang, Yong Wang, Michael P. Weiner, Pengguang Yu, Richard F. Begley, and Jonathan M. Rothberg. "Genome sequencing in microfabricated high-density picolitre reactors." *Nature*, **437**(7057):376–80, 9 2005.

[MFD10] Paul Medvedev, Marc Fiume, Misko Dzamba, Tim Smith, and Michael Brudno. "Detecting Copy Number Variation with Mated Short Reads." 2010. Genome Research.

[MH10] Jonathan Marchini and Bryan Howie. "Genotype imputation for genome-wide association studies." *Nat Rev Genet*, **11**(7):499–511, 06 2010.

[MHJ09] Ohtsuka M, Inoko H, Kulski JK, and Yoshimura S. "Major histocompatibility complex (Mhc) class Ib gene duplications, organization and expression patterns in mouse strain C57BL/6." 2009. BMC Bioinformatics.

[MHM07a] J. Marchini, B. Howie, S. Myers, G. McVean, and P. Donnelly. "A new multipoint method for genome-wide association studies by imputation of genotypes." *Nature genetics*, **39**(7):906–913, 2007.

[MHM07b] Jonathan Marchini, Bryan Howie, Simon Myers, Gil McVean, and Peter Donnelly. "A new multipoint method for genome-wide association studies by imputation of genotypes." *Nat Genet*, **39**(7):906–13, 7 2007.

[MMM08] John C. Marioni, Christopher E. Mason, Shrikant M. Mane, Matthew Stephens, and Yoav Gilad. "RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays." *Genome Research*, **18**(9):1509–1517, 2008.

[MMP09] V. Manquinho, J. Marques-Silva, and J. Planes. "Algorithms for weighted Boolean optimization." In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, pp. 495–508, 2009.

[MSB09] P. Medvedev, M. Stanciu, and M. Brudno. "Computational methods for discovering structural variation with next-generation sequencing." *Nature Methods*, **6**:S13–S20, 2009.

[MWM08] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. "Mapping and quantifying mammalian transcriptomes by RNA-Seq." *Nat Meth*, **5**(7):621–628, 07 2008.

[NSN05] Y. Nannya, M. Sanada, K. Nakazaki, N. Hosoya, L. Wang, A. Hangaishi, M. Kurokawa, S. Chiba, D.K. Bailey, G.C. Kennedy, et al. "A robust algorithm for copy number detection using high-density oligonucleotide single nucleotide polymorphism genotyping arrays." *Cancer research*, **65**(14):6071, 2005.

[PAG10] Bogdan Pasaniuc, Ram Avinery, Tom Gur, Christine F. Skibola, Paige M. Bracci, and Eran Halperin. "A generic coalescent-based framework for the selection of a reference panel for imputation." *Genetic Epidemiology*, 2010.

[PP09] Snehit Prabhu and Itsik Pe'er. "Overlapping pools for high-throughput targeted resequencing." *Genome Res*, **19**(7):1254–61, 7 2009.

[PPC08] K. Pipatsrisawat, A. Palyan, M. Chavira, A. Choi, and A. Darwiche. "Solving weighted Max-SAT problems in a reduced search space: A performance analysis." *Journal on Satisfiability, Boolean Modeling and Computation*, **4**:191–217, 2008.

[Pro10] 1000 Genomes Project. "A Deep Catalog of Human Genetic Variation. http://www.1000genomes.org/.", 2010.

[PS04] A. Panconesi and M. Sozio. "Fast hare: A fast heuristic for single individual SNP haplotype reconstruction." *Lecture Notes in Computer Science*, **3240**:266–277, 2004.

[Rep10] RepeatMasker. "http://www.repeatmasker.org/PreMaskedGenomes.html.", 2010.

[Sah09] S.C. Sahinalp. "Combinatorial Algorithms for Structural Variation Detection in High Throughput Sequenced Genomes." p. 218, 2009.

[SAZ10] Noam Shental, Amnon Amir, and Or Zuk. "Identification of rare alleles and their carriers using compressed se(que)nsing." *Nucleic Acids Res*, 8 2010.

[SB09] Can Alkan Seunghak Lee, Fereydoun Hormozdiari and Michael Brudno. "MoDIL: detecting small indels from clone-end sequencing with mixtures of distributions." *Nature Methods*, **29**, 2009.

[Sch08] Stephan C Schuster. "Next-generation sequencing transforms today's biology." *Nat Meth*, **5**(1):16–18, 01 2008.

[SCZ08] X. She, Z. Cheng, S. Z"ollner, D.M. Church, and E.E. Eichler. "Mouse Segmental Duplication and Copy-Number Variation." *Nature genetics*, **40**(7):909, 2008.

[SFD07] B.E. Stranger, M.S. Forrest, M. Dunning, C.E. Ingle, C. Beazley, N. Thorne, R. Redon, C.P. Bird, A. de Grassi, C. Lee, et al. "Relative impact of nucleotide and copy number variation on gene expression phenotypes." *Science*, **315**(5813):848, 2007.

[SLM07] J. Sebat, B. Lakshmi, D. Malhotra, J. Troge, C. Lese-Martin, T. Walsh, B. Yamrom, S. Yoon, A. Krasnitz, J. Kendall, et al. "Strong association of de novo copy number mutations with autism." *Science*, **316**(5823):445, 2007.

[SSD01] M. Stephens, N.J. Smith, and P. Donnelly. "A new statistical method for haplotype reconstruction from population data." *The American Journal of Human Genetics*, **68**(4):978–989, 2001.

[SSS08] Ian Sudbery, Jim Stalker, Jared T Simpson, Thomas Keane, Alistair G Rust, Matthew E Hurles, Klaudia Walter, Dee Lynch, Lydia Teboul, Steve D Brown, Heng Li, Zemin Ning, Joseph H Nadeau, Colleen M Croniger, Richard Durbin, and David J Adams. "Deep short-read sequencing of chromosome 17 from the mouse strains A/J and CAST/Ei identifies significant germline variation and candidate genes that regulate liver triglyceride levels." 2008. Genome Biology.

[TSB05] E. Tuzun, A.J. Sharp, J.A. Bailey, R. Kaul, V.A. Morrison, L.M. Pertz, E. Haugen, H. Hayden, D. Albertson, D. Pinkel, et al. "Fine-scale structural variation of the human genome." *Nature genetics*, **37**(7):727–732, 2005.

[VZC03] S. Volik, S. Zhao, K. Chin, J.H. Brebner, D.R. Herndon, Q. Tao, D. Kowbel, G. Huang, A. Lapuk, W.L. Kuo, et al. "End-sequence

profiling: sequence-based analysis of aberrant genomes." *Proceedings of the National Academy of Sciences*, **100**(13):7696, 2003.

[WGS09] Zhong Wang, Mark Gerstein, and Michael Snyder. "RNA-Seq: a revolutionary tool for transcriptomics." *Nat Rev Genet*, **10**(1):57–63, 01 2009.

[WLH07] K. Wang, M. Li, D. Hadley, R. Liu, J. Glessner, S.F.A. Grant, H. Hakonarson, and M. Bucan. "PennCNV: an integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data." *Genome Research*, **17**(11):1665, 2007.

[WSE08] D.A. Wheeler, M. Srinivasan, M. Egholm, Y. Shen, L. Chen, A. McGuire, W. He, Y.J. Chen, V. Makhijani, G.T. Roth, et al. "The complete genome of an individual by massively parallel DNA sequencing." *Nature*, **452**(7189):872–876, 2008.

[WWL05] R.S. Wang, L.Y. Wu, Z.P. Li, and X.S. Zhang. "Haplotype reconstruction from SNP fragments by minimum error correction." *Bioinformatics*, **21**(10):2456, 2005.

[ZGH09] F. Zhang, W. Gu, M.E. Hurles, and J.R. Lupski. "Copy Number Variation in Human Health, Disease, and Evolution." *Annual Review of Genomics and Human Genetics*, **10**:451–481, 2009.