Spring 5-7-2011

Inferring Genomic Sequences

Irina A. Astrovskaya
Georgia State University

INFERRING GENOMIC SEQUENCES

by

IRINA ASTROVSKAYA

Under the direction of Aleksandr Zelikovskiy

ABSTRACT

Recent advances in next generation sequencing have provided unprecedented opportunities for high-throughput genomic research, inexpensively producing millions of genomic sequences in a single run. Analysis of massive volumes of data results in a more accurate picture of the genome complexity and requires adequate bioinformatics support. We explore computational challenges of applying next generation sequencing to particular applications, focusing on the problem of reconstructing viral quasispecies spectrum from pyrosequencing shotgun reads and problem of inferring informative single nucleotide

polymorphisms (SNPs), statistically covering genetic variation of a genome region in genome-wide association studies.

The genomic diversity of viral quasispecies is a subject of a great interest, particularly for chronic infections, since it can lead to resistance to existing therapies. High-throughput sequencing is a promising approach to characterizing viral diversity, but unfortunately standard assembly software cannot be used to simultaneously assemble and estimate the abundance of multiple closely related (but non-identical) quasispecies sequences. Here, we introduce a new Viral Spectrum Assembler (ViSpA) for inferring quasispecies spectrum and compare it with the state-of-the-art ShoRAH tool on both synthetic and real 454 pyrosequencing shotgun reads from HCV and HIV quasispecies. While ShoRAH has an advanced error correction algorithm, ViSpA is better at quasispecies assembling, producing more accurate reconstruction of a viral population. We also foresee ViSpA application to the analysis of high-throughput sequencing data from bacterial metagenomic samples and ecological samples of eukaryote populations.

Due to the large data volume in genome-wide association studies, it is desirable to find a small subset of SNPs (tags) that covers the genetic variation of the entire set. We explore the trade-off between the number of tags used per non-tagged SNP and possible overfitting and propose an efficient 2LR-Tagging heuristic.

INDEX WORDS: Haplotype assembling, Viral quasispecies, Hepatitis C virus, Next generation sequencing, NGS, 454 pyrosequencing, Shotgun reads, VISPA, Read graph, Expectation maximization, Tagging, SNP, Genotype, Correlation, Linear Programming, Bioinformatics, Georgia State University

INFERRING GENOMIC SEQUENCES

by

IRINA ASTROVSKAYA

Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2011

INFERRING GENOMIC SEQUENCES

by

IRINA ASTROVSKAYA

Committee chair: Aleksandr Zelikovskiy

Committee: Yi Pan

Robert Harrison

Yury Khudyakov

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2011

ACKNOWLEDGMENTS

I would never have been able to finish my dissertation without the guidance of my committee members, love and support from my family and friends.

I would like to express my deepest gratitude to my advisor, Dr. Aleksandr Zelikovskiy, for his excellent guidance through my PHD life, unending patience and productive research atmosphere. It was a great pleasure to conduct a research under his guidance. I would like to thank my committee members Dr. Yi Pan, Dr. Robert Harrison and Dr. Yury Khudyakov for sharing their valuable knowledge with me and for challenging to do my best.

Special thanks go to my parents, Anatolii Astrovskii and Alexandra Astrovskaya, my sister Nataliya Lishyk, my niece Kseniya Lishyk and Nirajan Pathak for being a constant source of encouragement, support and love during my days in graduate school. They are always there cheering me up and stood by me through the good and bad times.

I want to thank my dearest friends Gulsah Altun, Dumitru Brinza, Kelly Westrbooks, Qiong Cheng, Vanessa Bertollini, Diana Mohan, and Serghei Mangul for reminding me that there is a life outside a school. I will always remember how much fun we have in our coffee club!

TABLE OF CONTENTS

LIST OF TABLES

rank of the assemblies. "OrigFreq" is frequency of the assemblies on the whole data. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found without repetition, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set cannot match the same assemblies obtained on 10%-reduced data. Note that table contains zero's average frequency. They are due to the precision: if the match was found only in one iteration and frequency was low, then after dividing by 100, it becomes 10^{-5} or less, and due to precision is shown as zero.

LIST OF FIGURES

1 INTRODUCTION

Recent advances in next generation sequencing have provided unprecedented opportunities for high-throughput genomic research, inexpensively producing millions of genomic sequences in a single run. Analysis of massive volumes of data results in a more accurate picture of the genome complexity by allowing researchers to explore a variety of underlying biological principals and mechanisms for organisms, whose genome was sequenced, however, requires adequate bioinformatics support. We explore computational challenges of applying next generation sequencing to particular applications, focusing on the problem of reconstructing viral quasispecies spectrum from pyrosequencing shotgun reads and problem of inferring informative single nucleotide polymorphisms (SNPs), statistically covering genetic variation of a genome region in genome-wide association studies.

A tagging problem can be viewed as an inherited subproblem of any genomic data analysis since, in practice, small percentage of allele values are missing. A missing allele value may occur either if this locus was not initially genotyped or if the value did not pass quality control threshold after typing. Then the initial preprocessing of the data is either ignoring this data point or imputing based on the known information in the dataset. In the latter case, one needs to solve a tagging problem and then impute missing values based on the obtained set of tag SNPs and prediction rule. For instance, we use imputation tagging-based algorithm in our pipeline for quasispecies assembling.

Both problems explore local genetic information to infer the global diversity of the data. Indeed, problem of inferring viral population searches connections between local fragments to restore underlying sequences, making “horizontal”-like inference. A tagging problem works in

"vertical" fashion by estimating genetic diversity between positions in DNA and removing those that at least decrease the available genomic variability in the data.

From computer science perspective, two problems are very similar since the main computation challenge is to concisely represent the space of all feasible solutions. Graph-theoretic approach usually allows not only capture all information but also effectively traverse plausible solutions.

1.1 Reconstruction of HCV Quasispecies Haplotypes

In epidemiology, RNA viral genomes (for instance, Hepatitis C virus (HCV), Human Immunodeficiency Virus (HIV)) are a subject of the great interest since the exact mechanisms of their persistence and evolution remain open questions. HCV is an important human pathogen that induces a chronic infection in 3% of the world's population. It primarily infects the liver and the majority of infected subjects develop progressive liver disease. As many RNA viruses, HCV exists as a set of closely related sequences, i.e., quasispecies. The diversity of genomic sequences in an infected individual can cause the failures of vaccines and virus resistance to existing therapies. Knowing sequences of the most virulent variants can help to design effective drugs [4, 24] and vaccines [10, 15] targeting particular viral genome *in vivo*.

Ultra-deep sequencing, provided by 454 Life Sciences system, is a promising technology for viral quasispecies analysis. Since existing high-throughput sequencing systems are originally designed for a single genome assembly, they cannot readily distinguish and simultaneously assemble multiple closely related sequences as well as estimate their relative abundances. Hence, new software tool should be developed to assemble sequenced reads into multiple closely related (but non-identical) sequences.

We propose a novel algorithm Viral Spectrum Assembler (ViSpA) for inferring viral quasispecies spectrum from 454 pyrosequencing shotgun reads. The method takes into account sequencing errors at multiple steps, including mapping-based read preprocessing, path selection based on maximum-bandwidth, and candidate assembly using probability-weighted consensus techniques. ViSpA also considers sequencing errors in its Expectation Maximization (EM) algorithm for frequency estimation of quasispecies sequences.

ViSpA has been compared with the state-of-the-art ShoRAH on both synthetic and real 454 pyrosequencing shotgun reads from HCV and HIV-1 quasispecies. Experimental results show that ViSpA enables more accurate inferring of viral quasispecies spectrum reconstruction from 454 pyrosequencing reads.

1.2 Genotype Tagging with Limited Overfitting

DNA variations, primarily single nucleotide polymorphisms (SNPs), hold many promises as a basis of the genome-wide association studies. Genome-wide studies are challengeable due to the genotyping cost and computational complexity of the analysis. Since allele values of neighboring SNPs are usually correlated, a smaller subset of so called informative (tag) SNPs can represent (cover) the entire genetic variation of the set of SNPs, thus, can be used as a basis of genome-association studies. Informally, the problem of tagging is selecting a minimum number of tag SNPs covering all SNP in the given data.

To represent genetic variation of an untagged SNP, some tagging methods use a single tag (IdSelect [11] and Tagger [15]), or multiple tags (MLR [29]). If a single tag used to cover the genetic variation of untagged SNP, the tagging problem can be efficiently approximated and almost the optimum number of tag SNPs is found. If multiple tags can be used per untagged SNP, there is no efficient approximation for the tagging problem although a smaller subset of tag

SNPs is obtained. In case of MLR method, gain in number of tag SNPs leads to data overfitting since multiple unrelated tags can participate in covering of an untagged SNP.

Here, we explore the trade-off between the number of tags and overfitting and considers the problem of finding a minimum number of tags when at most two tags can represent variation of an untagged SNP. We show that this problem is hard to approximate and propose an efficient heuristic, referred as 2LR. The experiments show that 2LR is between Tagger and MLR in the number of tags and in overfitting. Indeed, 2LR uses slightly more tags than MLR but the overfitting measured with 2-fold cross validations is practically the same as for Tagger.

1.3 Roadmap

The remainder of the dissertation is organized as follows. Chapter 2 presents the Quasispecies Spectrum Reconstruction problem: its motivation, formulation, our algorithmic approach, proposed validation methods as well as experimental results on simulated data and 454 pyrosequencing shotgun reads from HCV and HIV samples. Chapter 3 presents the Minimum Tag Selection Problem, beginning with the problem motivation, formulation, exact and approximate solutions and experimental results on publicly available datasets. Conclusions and future work is described in the Chapter 4.

2 INFERRING VIRAL QUASISPECIES SPECTRUM FROM 454 PYROSEQUENCING READS

RNA viruses infecting a host usually exist as a set of closely related sequences, referred to as quasispecies. The genomic diversity of viral quasispecies is a subject of the great interest since it is a plausible cause of vaccines failures and virus resistance to existing therapies. High-throughput sequencing is a promising approach to characterize viral diversity; however, standard assembly software was originally designed for a single genome assembly and cannot be used to simultaneously assemble multiple closely related sequences and estimate their relative abundances, in the other words, reconstruct the quasispecies spectrum. In this chapter, we introduce a novel **Vi**ral **Sp**ectrum **A**ssembler (ViSpA) method for inferring viral quasispecies spectrum from 454 pyrosequencing shotgun reads.

2.1 Viral Quasispecies

Many medically important viruses, including SARS coronavirus (SARS-CoV), influenza viruses, Hepatitis B virus (HBV), Hepatitis C viruses (HCV), and Human Immunodeficiency virus (HIV), encode their genome in RNA. Unlike DNA, RNA lacks ability to detect and repair mistakes during replication since RNA polymerase is unable to perform the proof reading of DNA polymerase associated with $3' \rightarrow 5'$exonuclease activity [21]. As a result, RNA virus mutation rate can be as high as 1 mutation per each 1000-100000 bases copied per replication cycle [20]. To ensure its survival, RNA virus exploits various mechanisms of sequence variation including substitution, insertion, deletion, recombination and genome segment reassortment. Many of the mutations are well tolerated and passed down to descendants, producing a family of

co-existing related (but non-identical) variants of the original viral genome referred to as ***quasispecies***, a concept that originally described a mutation-selection balance [18, 41, 48].

Short replication cycle, small size of genome, immune response of an infected patient additionally contribute to the evolution of the quasispecies, resulting in 10^{10} to 10^{12} new variants per day in a host infected with HIV-1, HBV, or HCV [19]. These quasispecies variants differ in their biological properties such as virulence, or ability to cause disease, ability to escape the immune system, resistance to antiviral therapies, and tissue tropism, or cells and tissues of a host supporting growth of virus. As a consequence, the best adapted variants survive different environmental changes and selective pressures, over time, making viral quasispecies spectrum different in simultaneously infected hosts.

The diversity of viral sequences and their frequent generation in an infected individual can cause vaccines failures and virus resistance to existing therapies, resulting in just few RNA viruses being effectively controlled by vaccination or antiviral therapies [19, 31]. Indeed, it is extremely difficult to control chronic active viral hepatitis. For example, care therapy for HCV-1 infected patient is effective only in 50% of cases, however, has many side effects ranging from a flu-like symptoms to serious adverse events such as damage of heart muscle and anemia. Furthermore, quasispecies nature of RNA viruses is a challenge in vaccine development because live viruses rapidly mutate and potentially become virulent for a host. Therefore, there is a great interest in reconstructing genomic diversity of viral quasispecies in an infected host. Knowing sequences of virulent variants and their abundance in infected patients can improve our understanding of viral evolutionary dynamics, mechanisms of viral persistence and drug resistance, and, as result, may help to design an effective drugs and treatments targeting particular viral genome *in vivo*.

2.2 Roche 454 Pyrosequencing Technology

Roche 454 GS FLX Titanium is a next-generation, massively-parallel pyrosequencing instrument designed for DNA sequencing [1]. Unlike Sanger, ultra deep pyrosequencing detects mutations at extremely low levels, allowing discovery of rare drug-resistant viral quasispecies.

Briefly, the 454 pyrosequencing system shears the source genetic material into fragments of approximately 300-800 bases called ***reads*** (see Fig. 2.1). Then two short adaptors are attached to the ends of each read, and DNA capture beads bound single-stranded reads, one DNA molecule per bead. At each bead, PCR amplification occurs in an emulsion with amplification reagents in a water-in-oil mixture, resulting in many copies of the single-stranded DNA fragment. Then beads are deposited into individual PicoTiterPlate wells in a fiber optic slide, and millions of reads are sequenced by synthesizing their complementary strands. Repeatedly, nucleotide reagents are flown over the read in a fixed order, one nucleotide (A, C, T, or G) at a

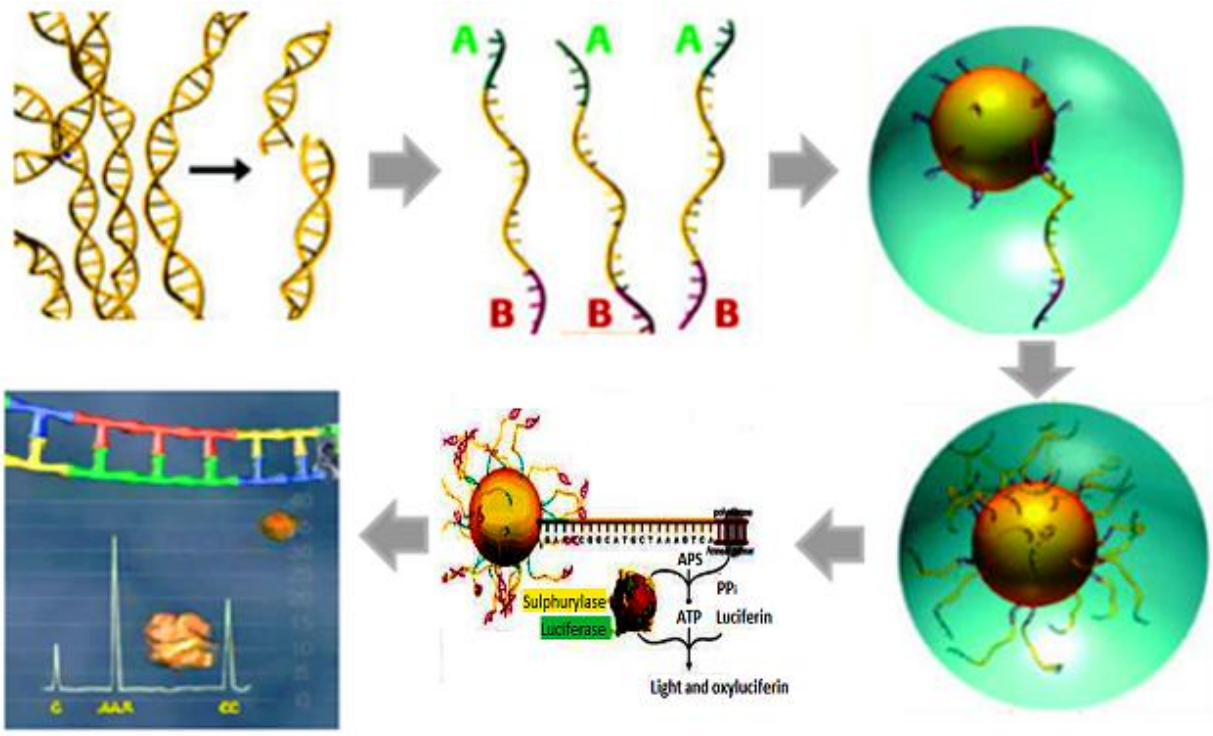

Figure 2.1 *Pyrosequencing using Roche 454 Life Science platform [1].*

time. Light is emitted when the flown nucleotide base complements the first unpaired base of the fragment [23, 40]. Sequence of chemiluminescent signals is recorded as flowgram by camera and later translated into reads sequences.

The well-known drawback of 454 pyrosequencing is in length resolution of a ***homopolymer***, that is, sequence of identical bases (e.g., AAAAAA). During pyrosequencing, multiple identical nucleotides may be incorporated in a single cycle, and, ideally, light intensity corresponds to the number of incorporated bases. However, in reality, light intensity varies substantially for the same length of homopolymer, and incorrect resolution causes nucleotide over-calls and under-calls. The resulted insertions and deletions respectively constitute 65%-75% and 20%-30% of all sequencing errors [4, 9].

Finally, the original genome is reconstructed computationally. Since the system was originally designed to sequence genetic material from a single species, the standard software assembles all reads to a single consensus genome, and cannot be used for reconstructing quasispecies sequences.

The Roche 454 GS FLX Titanium can sequence around 400-600 million filter-passed bases and generate more than 1 million high-quality reads with average read length up to 400 bases per a single 10-hour run [1]. In future, the average read length may be as large as 750 bases.

2.3 Quasispecies Spectrum Reconstruction Problem

The goal of viral assembling software is to correctly infer (reconstruct) distribution of viral closely related quasispecies in the sequenced sample, or in the other words, it addresses the following problem.

Quasispecies Spectrum Reconstruction (QSR) Problem: *given a collection of 454 pyrosequencing shotgun reads generated from a viral sample, reconstruct the quasispecies spectrum, i.e., the set of sequences and the relative frequency of each sequence in the sample population*.

Major challenge in solving the QSR problem is due to the slight differences among quasispecies sequences and relatively high sequencing error rate in data generated by current technologies. Higher diversity among quasispecies sequences helps to identify more sequencing errors as "outliers" and easier partition reads among individual quasispecies. When a viral population has low diversity, sequencing errors may have the same frequency as rare mutants in the population, making some quasispecies sequences indistinguishable from each other. The amount and distribution along the genome of differences between quasispecies significantly varies between virus species, since different species have different mutation rates and genomic architectures. In particular, due to the lower mutation rate and longer conserved regions, HCV quasispecies are harder to reconstruct than quasispecies of HBV and HIV. Additionally, the QSR problem is made difficult by the limited read length. Indeed, the shorter reads are, the more feasible partitions are available, and larger chance to assemble *in-silico* (not real) sequences. Longer reads impose additional constraints on reads partitioning and lead to more accurate solutions.

The QSR problem is related to several well-known problems, including *de novo* genome assembly (see, for example, [10, 13, 17, 32, 35, 42, 43, 50, 53, 59]), haplotype assembly (see, for example, [7, 14, 25, 38, 39, 45, 56]), population phasing (see, for example, [8]) and metagenomics (see, for example, [34, 36, 44, 52]). As noted above, *de novo* assembly methods are designed to reconstruct a single genome sequence and are not well-suited for reconstructing a

large number of closely related quasispecies sequences. One can reduce QSR problem to *de novo* genome assembly if all quasispecies sequences are placed one-by-one in a long sequence and common segments are treated as repeats. Then the fragments (reads) should be assembled into the single genome. However, *de novo* genome assembly is NP-hard problem due to the presence of repeats, and no efficient solution is available: an assembler should either guess the correct genome among many alternatives or produce a fragmented assembly by restricting itself to non-repetitive segments of the genome. Further, QSR problem reminds the haplotype assembly that seeks to reconstruct two closely related haplotype sequences of the same diploid genome. Unfortunately, existing methods are not scalable with respect to the number of sequences which is also *a priori* unknown. Computational methods developed for population phasing deal with large numbers of haplotypes, but rely on the availability of genotype data that conflates information about pairs of haplotypes. Finally, viral quasispecies sequencing can be viewed as sequencing of environmental samples, or metagenomics, when obtained reads are sequenced from DNA mixture of many different species. However, the differences between the genomes of these species are considerably larger than those of viral quasispecies. Furthermore, existing tools for metagenomic data analysis focus on species identification, as reconstruction of complete genomic sequences would require much higher sequencing depth than that typically provided by current metagenomic datasets. In contrast, achieving high sequencing depth for viral samples is very inexpensive, owing to the short length of viral genomes.

Mapping based approaches to QSR are naturally preferred to *de novo* assembly since reference genomes are available (or easy to obtain) for viruses of interest (for instance, from NCBI [3]), and viral genomes do not contain repeats. Thus, it is not surprising that such approaches were adopted in the two pioneering works on the QSR problem [22, 54]. Both

approaches build a read graph, where vertices correspond to non-redundant aligned error-free reads, edges represent overlaps between reads, and possible quasispecies sequences are paths from the leftmost to the rightmost vertices in the graph.

Eriksson et al. [22] proposed a multi-step approach consisting of sequencing error correction via local deterministic clustering, haplotype reconstruction via chain decomposition, and haplotype frequency estimation via expectation maximization (EM) method. The proposed method was able to reduce sequencing error rate by factor of 30 on simulated data and infer (within one amino acid difference) at least 51.8% of HIV-1 population from 5,177 reads with average read length 105 bases. The error correction of this approach was further improved by Zagordi et al. [57]. The authors applied Dirichlet process mixture model for probabilistic local clustering and treated each base accordingly to majority rule in 3 consecutive windows. The method was implemented in assembling software ShoRAH and validated in HIV-1 population [58]. ShoRAH was able to detect haplotypes at frequencies at least 0.1%, and distances between reconstructed and original haplotypes were always below 1%.

In Westbrooks et al. [54, 55], the focus was on haplotype reconstruction via transitive reduction, overlap probability estimation and network flows application. The method was validated on error-free reads simulated from 1,739bp-long fragment of E1E2 region of 44 HCV quasispecies. In this dissertation, we have further extended this approach to the case of real 454 pyrosequencing reads from a viral sample.

Recently, Prosperi et al. [46] introduced a novel greedy algorithm minimizing the number of reconstructed *in-silico* recombinants by coupling overlapping reads with similar frequencies across the various amplicons, overlapping regions, covering the entire genome. The

combinatorial algorithm was applied to simulated data and real reads from HBV quasispecies, showing similar to ShoRAH results.

For simplicity of solution description, we further refer QSR problem as two following problems.

Quasispecies Sequences Assembly (QSA) Problem: *assemble the set of quasispecies sequences that is consistent with 454 pyrosequencing shotgun reads generated from a viral sample and the reference (consensus) sequence.*

Quasispecies Frequencies Estimation (QFE) Problem: *estimate frequencies of the given sequences to be consistent with the 454 pyrosequencing shotgun reads.*

As noted above, we have extended the approach in [54, 55] for real 454 pyrosequencing reads produced from a viral sample. The contributions of the dissertation include

- novel QSR tool called **Vi**ral **Sp**ectrum **A**ssembler (ViSpA) consisting of
  - ✓ aligned reads preprocessing including a simple error correction,
  - ✓ efficient solution to QSA problem via maximum-bandwidth path selection and mutation-based clustering for haplotype assembling in weighted read graph,
  - ✓ solution to QFE problem via maximum-likelihood estimation, taking into account all reads in the data;
- comparison of ViSpA with the state-of-the-art ShoRAH on HCV synthetic data both with and without sequencing errors;
- statistical and experimental validation of the two methods on real 454 pyrosequencing reads from intra-host HCV and HIV samples.

The following sections describe ViSpA's approach to QSA problem (see Section 2.4.1) and EM-based algorithm to solve QFE problem (see Section 2.4.2).

2.4 Viral Spectrum Assembler (ViSpA)

Our method ViSpA assembles viral quasispecies sequences from 454 pyrosequencing shotgun reads by considering sequencing errors at multiple steps, selecting paths corresponding to the most probable quasispecies sequences, and assembling candidate sequences based on an expected mutation rate in a viral sample (see Section 2.4.1). Then EM-based algorithm finds maximum likelihood frequency estimation for each assembled sequence, based on all 454 pyrosequencing shotgun reads (see Section 2.4.2).

2.4.1 Assembly of Quasispecies Sequences

ViSpA solves QSA problem as follows (see Figure 2.2).

1. Align 454 pyrosequencing reads either versus reference or consensus constructed for the given sample (for details, see Section 2.4.1.1).
2. Preprocess aligned reads to correct sequencing errors (for details, see Section 2.4.1.3).
3. Construct a transitively reduced read graph with vertices representing reads and edges representing overlaps with partial agreement between overlapped reads (for details, see Section 2.4.1.4).
4. Select most probable paths per each vertex, and assemble candidate sequences for selected paths by weighted consensus of reads (for details, see Section 2.4.1.5).

Below we describe each step separately.

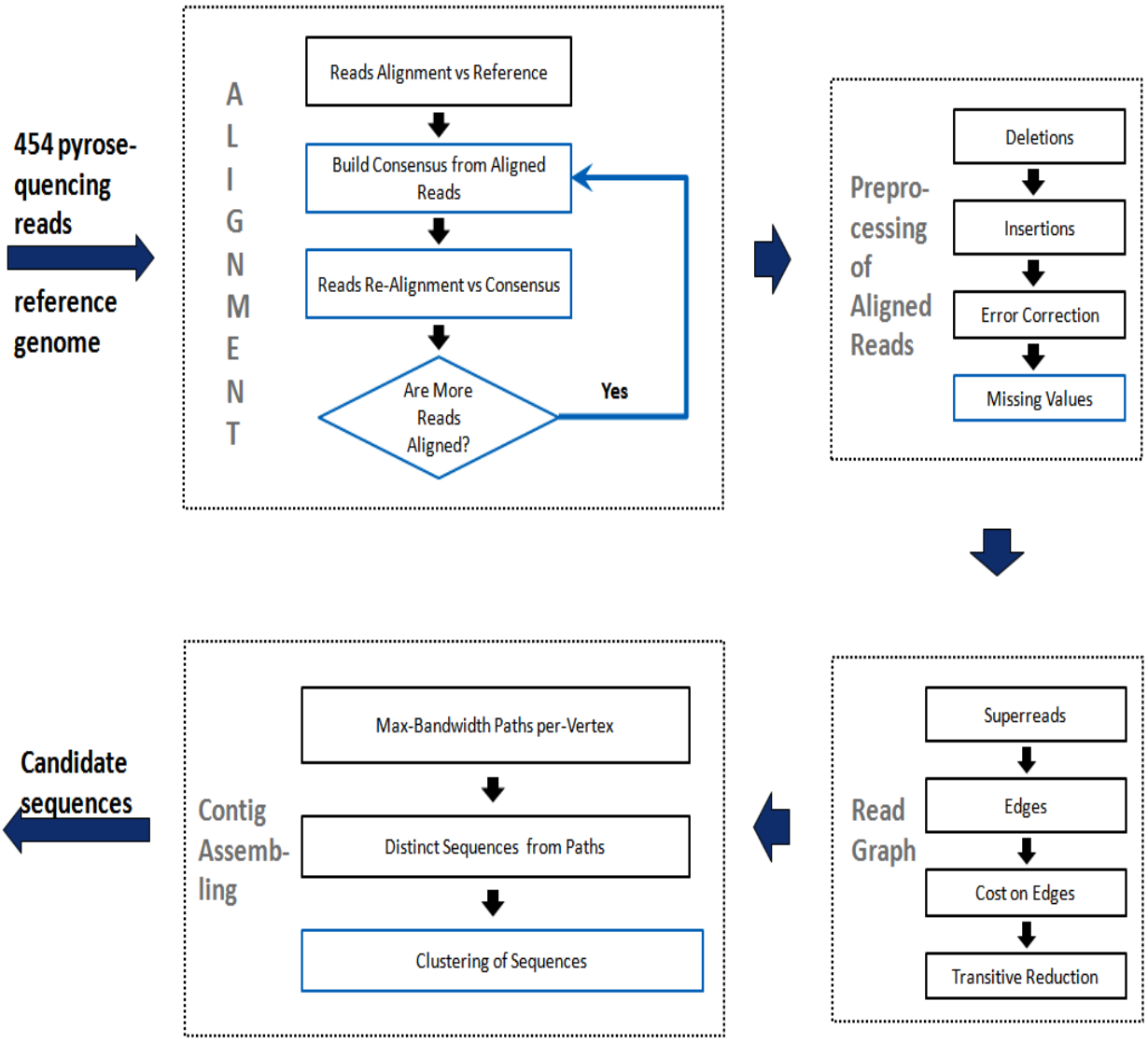

Figure 2.2 *ViSpA's flow for QSA problem. Optional steps in ViSpA's viral quasispecies assembling are in shapes with blue borders and include consensus construction and iterative realignment to consensus, preprocessing of missing values and clustering of candidate sequences.*

2.4.1.1 Read Alignment and Consensus Construction

Viral quasispecies assembling starts with read alignment wherein starting position of each read is determined relatively to either a reference or constructed consensus sequences. We assume that a reference genome sequence of the particular viral strain is publicly available (e.g., from NCBI [3], HIV databases [2]). Since viral genomes do not have sizable repeats and the

quasispecies sequences are usually close enough to the reference sequence, in general, majority of reads can be uniquely aligned onto the reference genome. However, a significant number of reads may remain unaligned due to the differences between the reference genome and sequences in the viral sample. These differences are caused by high mutation rate and viral evolution guided by an individual response of host's immune system. In order to recover as many of these reads as possible, we can iteratively construct a consensus genome sequence from the already aligned reads.

In particular, we first align 454 pyrosequencing shotgun reads to the reference sequence using the SEGEMEHL software [49]. The alignment score should satisfy two conditions. First, the lower score reflects the smaller number of differences between aligned read and the reference sequence. And secondly, the alignment score penalizes mismatches less than insertions into a read and deletions from a read with respect to the reference sequence. Then we extend the reference sequence with a placeholder *I* for each nucleotide inserted by at least one uniquely aligned read. Similarly, we add a placeholder *D* to a read sequence for each reference nucleotide missing from the aligned read. Next we perform sequential multiple alignment of the previously aligned reads against this extended reference sequence. Finally, the consensus sequence is obtained by (1) replacing each nucleotide in the extended reference with the nucleotide or placeholder in the majority of the aligned reads and (2) removing all *I* and *D* placeholders respectively corresponding to rare insertions and deletions found in a majority of the reads. Reads may contain a small percentage of unidentified nucleotides denoted by *N*'s. We treat *N* as a special allele value matching to any of nucleotides *A, C, T, G* as well as placeholders *I* and *D*.

Iteratively, we replace the reference with the consensus and try to align the reads, for which we could not find any acceptable alignment previously. Our experiments on a dataset,

consisting of approximately 31,000 454 pyrosequencing shotgun reads generated from 5.2kb-long HCV fragment (see data description in Section 2.6.2), show that 85% of reads are uniquely aligned onto the reference sequence and an additional 9% of reads are aligned onto the final consensus sequences. Reads that cannot be aligned onto the final consensus are removed from further consideration.

2.4.1.2 Genotyping Errors

Although next generation sequencing techniques can generate significantly more sequence information at a lesser cost than Sanger sequencing, their sequencing accuracies are much lower of Sanger's. For example, 454 GS FLX instrument may erroneously sequence up to four out of 1,000 bases due to the technology limitations [4].

As noted above, a well-known drawback of 454 pyrosequencing is a large amount of erroneous ***indels*** [4, 9], i.e., insertions into reads and deletions from reads with respect to a reference (consensus) sequence, caused by incorrect resolution of homopolymer length**.** During pyrosequencing, nucleotides are repeatedly added in the fixed order over the wells and light intensities are recorded into flowgram (see Figure 2.3). The rounded value of each light signal is the length of the corresponding homopolymer. For instance, flows in Figure 2.3 would be interpreted as *TAAAGGCCG* sequence. Indeed, rounded values for the flown in the first cycle nucleotides *T, A, C* and *G* are 1, 3, 0, and 2, respectively. The second cycle adds only two *C*'s and one *G* to the sequence since light intensities for *T* and *A* are under 0.5.

Ideally, light intensity should be proportional to the correct length of a homopolymer. However, signal for the same length varies and sequencing error occurs when the difference

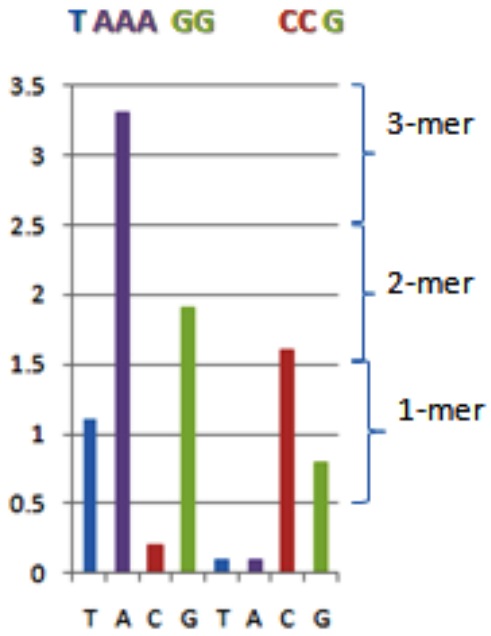

Figure 2.3 *Example of 454 flowgram. Nucleotides are flown in the order T, A, C, G. The light intensity intervals on the right are the lengths of homopolymers. The sequenced read - TAAAGGCCG - is shown above the flowgram.*

between signal intensity and the true value exceeds 0.5. As a consequence, most of the sequencing errors are over-calls (65%-75% of sequencing errors are insertions), or under-calls (20%-30% of errors are deletions). Furthermore, the indels may lead to incorrect read alignment, increasing number of substitution sequencing errors (around 5% of sequencing errors). For example, assume sequencing *TAAACGGCG* fragment produces the flowgram on Figure 2.3. Since *C* nucleotide has inadequate intensities in both cycles (too low and too high light intensities in the first and the second cycles, respectively), sequencing platform interprets flowgram as *TAAAGGCCG* sequence. Since mismatch is penalized less than indels during alignment, the substring, starting at the 5^{th} nucleotide in the read (*GGCCG*), is incorrectly aligned onto *CGGCG* in the reference genome, failing to recognize deletion at the 5^{th} position and insertion at the 8^{th} position and introducing two substitution errors.

In case of viral quasispecies assembling, handling sequencing errors is challenging. Indeed, sequencing error rate makes an infrequent allele value and a sequencing error indistinguishable and increases chance of considering technical error as a low-frequency viral variant since quasispecies sequences may differ as little as one nucleotide. Nevertheless different statistical methods were proposed to improve quality of base calls [1, 4, 9], more than half of the reads would be contaminated with at least one sequencing error with average read length of 350 bases [58]. Different strategies were proposed to decrease number of sequencing errors such as removing low-quality reads, that is, reads having at least one nucleotide with quality score less than 10, or reads that are considerably shorter than the average, or reads that result in frameshifts in the translation [58]. More sophisticated approaches to error detection/correction problem include considering less frequent bases in multiple sequence alignment as errors [58, 61, 62]; removing errors based on specific graph structures such as short dead-end paths, multiple paths between two nodes, and paths with low multiplicity in de Bruijn graph [59]; distinguishing between erroneous k-mers and correct k-mers based on k-mers count profile in data and applying the minimum number of corrections to erroneous k-mers [10, 13, 63, 64].

In this work, we assume that every position in each quasispecies is covered by multiple reads. Since it is extremely unlikely to have the same sequencing error at the same position in two reads produced by the same quasispecies, we can disregard information that is supported only by a single read. Such naïve and conservative error correction removes obvious technical noise without removing essential information for reconstructing low-diversity viral population and without introducing additional noise due to correction procedure.

2.4.1.3 Aligned Reads Preprocessing

Since aligned reads may contain true insertions and deletions, sequencing errors and missing values, we (1) use placeholders *I* and *D* to simplify position referencing among the reads, (2) detect and correct obvious technical noise, and, (3) optionally, impute unidentified values in the reads.

First, we substitute each deletion in the aligned reads with placeholder *D* and use placeholder *I* for each position in the reads with a gap corresponding to the insertion in the other read (see Figure 2.4 (left)). All placeholders are treated as additional allele values but they are

Figure 2.4 *Aligned reads preprocessing. Left: example of indels preprocessing. At the top, multiple reads alignment is given. At the middle, I and D placeholders are added. At the bottom, deletions, supported by a single read, are corrected, insertions, confirmed by a single read, are removed. Right: example of imputing missing value N in a read. Let i be a position with a missing value N. Since the read has T at the position i - 1 and has no A at the position i + 1, A is imputed.*

removed from the final assembled sequences. In our initial set of experiments on HCV dataset (see Section 2.7.2.1), we also used to remove all insertions from the data.

Next we apply naïve error detection/correction procedure (see Algorithm 1) to fix obvious sequencing errors in the read data under assumption of multiple coverage of each position at every quasispecies.

Algorithm 1 Naïve error detection/correction method.

Input

1. $R = \{r_i\}_{i=1}^{M}$ is the set of aligned reads sorted in ascending order of their starting positions.
2. *ref* is an extended reference (consensus) sequence of a length *L*.

❖ $corR = \{cor\_r_i\}_{i=1}^{M1}$ is the set of corrected reads (in ascending order of their starting positions).

❖ *profile[L+1]* stores profile for each position, including

- how many reads, covering the position, support each allele value, placeholders and missing value (initially, set to 0),
- flags that determine if position should be removed or corrected (initially, set to false),
- and if corrected, which allele value is correct,
- to which position map after removing erroneous insertions (initially, set to -1).

corR ← ∅

for each read *r* in *R* **do**

    for each position *i* covered by *r* **do**

        update corresponding count at *profile[i]* based on the symbol in *r* at the position *i*

    end for

end for

pos ← 0

for each position *i* from 1 to *L* **do**

total is the number of reads covering *i*

if count for *I* or *D* in *profile [i]* is *total* or *total* -1 **then**

profile[i].toRemoveFlag ← *TRUE*

continue

end if

if count for *D* in *profile [i]* is 1 **then**

profile[i].toCorrectD ← TRUE

profile[i].dTo is allele value for which count is *total* -1, or *N*.

end if

if there is count for some allele in *profile [i]* equaled to *total* -1 **then**

profile[i].toCorSub ← TRUE

profile[i].toCor is allele value for which count is *total* -1

end if

pos ← *pos*+1

profile[i].mapTo ← *pos*

end for

update *ref* by removing all positions for which profile[] has *toRemoveFlag* set to TRUE

for each read *r* in *R* **do**

pos ← *r.beginning_position*

while *pos* < *L*+1 AND *profile[i].toRemoveFlag* is TRUE **do**

pos ← *pos +1*

end while

if *pos* is *L*+1 **then**

continue

```
        end if
        seq ← empty string
        for each position i=pos to r.endingPosition do
            if profile[i].toRemoveFlag is TRUE then
                continue
            end if
            if profile[i].toCorrectD then
                seq ← seq + profile[i].dTo
            else if profile[i].toCorSub then
                seq ← seq + profile[i].toCor
            else   seq ← seq + symbol in r at the position i
            end if
        end for
        corR ←corR U {read with sequence seq, starting at position profile[pos].mapTo}
end for
return corR, ref
```

Each deletion, supported by a single read, is replaced either with the allele value, which is present in all other reads, overlapping this position, or with *N*, signifying an unknown value, otherwise. All insertions, supported by a single read, are removed from further consideration. Finally, each allele value, supported by a single read, is replaced by the allele value, on which all other reads, covering this position, agree, or is left, if several allele values are introduced at the position by the reads. For example (see Figure 2.4 (left)), since there is no agreement on allele value at the third position, deletion in the *Read1* is replaced with *N* whereas deletion in *Read2* should be replaced by *G*. Obviously, insertion in the *Read3* is a sequencing error whereas one-base insertion in the *Read2* may be a true insertion, thus, should be kept for further analysis.

In general, the runtime of Algorithm 1 is $O(\#reads \cdot \bar{l} + L)$, where $\bar{l}$ is average read length in the data, and L is the length of extended reference (consensus) sequence. In practice, runtime of Algorithm 1 can be bounded by $O(\#reads \cdot \bar{l})$ due to much smaller size of viral genomes in comparison of 454 GS FLX throughput.

The experiments on HCV dataset show that even such conservative error correction of indels significantly improves the quality of assembled sequences, decreasing number of candidate sequences with stop codons in the coding region of the corresponding amino-acid sequences. Although the last step of the method – correction of substitution errors - can be skipped since ViSpA effectively correct most of substitution errors during assembling of candidate sequences. Additionally, Algorithm 1 neither introduces any additional noise due to incorrect error resolution nor reduces underlying diversity of the data, allowing ViSpA to correctly infer more viral quasispecies sequences from 454 pyrosequencing shotgun reads generated from low-diversity viral population than from the same reads, being corrected by ShoRAH, for example. If data are too noisy, one can run ViSpA after detecting and correcting technical errors by available software (see Section 2.4.1.2).

At the final step, we either impute unidentified missing values or treat *N* as a special allele value that matches to any of nucleotides *A, C, T, G* or placeholders *I* and *D*.

Our imputation method is based on the notion that allelic values at neighboring positions are usually highly correlated. Tagging methods (for more details, see Section 3) statistically measures the correlation between position of the interest and any of its neighbors to identify those that can either cover genetic variation at the current position, or allow to infer (predict) possible missing value based on the underlying genotype/haplotype data. The correlation of at

least 80% is expected for reliable inference. Hence, our imputing method can find highly (statistically) correlated neighbors (so called ***tags***) for each position where missing value occurs and then predict it as an allele value which is either present in the most associated tag, or followed by a majority rule on all associated tags. For instance (see Figure 2.4 (right)), let *i* be a position with a missing value *N* in a given read. Based on available data, *T* at the position *i - 1* 80% of times predicts *A* at the i^{th} position whereas *A* at the position *i + 1* 80% of times predicts *C* at the position *i*. Since our read has *T* at *i+1* position and has no *A* at *i+1* position, *N* is replaced with *A* as the allele value which is present in the most associated tag.

In contrast to genome-wide disease association studies, calculation of correlation is not straightforward in the case of viral quasispecies sequences. Indeed, correlation calculation assumes at most two different allele values at a particular position; however, all four allele values can be found in a position across viral quasispecies sequences. In the later case, we suggest to measure correlation between two positions as ability of specific allele value in one position to infer particular allele value in the other with considerably high (for example, more than 80%) confidence. In the other words, for each pair of positions *i* and *j*, we consider all 16 haplotypes of the length 2 (placeholders *I* and *D* are not considered). Let X_iY_j be one of these haplotypes with X_i and Y_j being an allele values in the positions *i* and *j*, respectively ($X_i, Y_j \in \{A, C, T, G\}$). Then SNPs at the positions *i* and *j* are binary coded as follows:

$$\mathrm{Z} \rightarrow \begin{cases} I_{X_i}(Z), & \text{at i-th position,} \\ I_{Y_j}(Z), & \text{at j-th position,} \end{cases} \tag{2.1}$$

where $Z \in \{A, C, T, G\}$ is any allele value in the positions *i* and *j* and $I_{X_i}(\cdot)$ and $I_{Y_j}(\cdot)$ are two indicator functions. For instance, haplotype X_iY_j is *AC* then all *A*'s at the position *i* and all *C*'s at

the position *j* are coded as 1's and all other values at both positions are coded as 0's. Then correlation is calculated by the standard formula for two binary vectors (see Section 3).

Allele X_i in a position *i* and allele Y_j in a position *j* are called a **tag box for the position j** if the correlation between corresponding binary vectors are greater than 0.8. In general, existence of X_i Y_j tag box for the position j does not imply existence of Y_j X_i tag box for the position *i*.

Formally, our imputation method addresses two problems.

Informative Tag Box Selection (ITS) Problem: *given a sample of a population S of n haplotypes, each consisting of m alleles, and index i, $i \leq m$, find tag boxes for position i.*

Tagging Based Prediction (TBP) Problem: *given a sample of a population S of n haplotypes, each consisting of m alleles, a haplotype h with unknown value at the position i, $i \leq m$, and a set of tag boxes for i^{th} position, predict allele value at the position i of the haplotype h.*

Algorithm 2 Imputation of missing values *N*.

Input

1. *R* is the set of reads without *N*'s sorted in ascending order of their starting positions.
2. *I* is the sorted set of positions at which at least one read has *N*.
3. R_I is the set of reads with at least one *N* sorted in ascending order of their starting positions.

❖ *S[|I|]* stores list of tag boxes per each position in I

for each position *i* in *I* **do**

get reads from $R \cup R_I$ covering entire window *w* (by default, width of window is set to 31)

for position *k* in window *w* **do**

if *k! = i* **then**

for each haplotype X_kY_i in {*AA, AC, AT, AG . . . GG*} **do**

binary code positions *k* and *i* by the formula (2.1)

if $r_{XkYi} \geq 0.8$ **then**

add to the list of tag boxes *S[i]*, tag box: (k, X_kY_i, $cor = r_{XkYi}$)

end if

end for

end if

end for

end for

for each position *i* in *I* **do**

$R_i \in R_I$ is the set of reads that have *N* at position *i*

$cur[|R_i|]$ stores a tag box per each $read_k$ in R_i

for each tag box *T: (j, X_jY_i, $cor=r_{XjYi}$)* in *S[i]* **do**

for each read $read_k$ in R_i **do**

if $read_k$ has X_j at position *j* AND *($cur[read_k]$ is NULL OR T.cor >$cur[read_k].cor$)*

then $cur[read_k] \leftarrow T$

end if

end for

end for

for each read $read_k$ in R_i **do**

Replace *N* in the $read_k$ with X_j value in $cur[read_k]$

if $read_k$ has no *N* **then**

$R \leftarrow R \cup \{ read_k \}$

Remove $read_k$ from R_I

end if

end for

end for

return *R*

In the general case, ITS problem requires to traverse $16n(n-1)$ tag boxes for n positions. Since correlation is typically decreasing with larger distance between positions, we solve ITS problem locally, in narrow window, for example, of width 31 bases. Then the worst-case runtime for ITS problem is $O(16 \cdot |I| \cdot w \cdot MaxReadCoverage)$ where $|I|$ is the number of positions at which at least one read has missing value *N*, and *w* is a width of the used window. The worst-case runtime for TBP problem is $O(4 \cdot \#N \cdot w)$. Thus, worst-case runtime for Algorithm 2 can be bound by $O(\#N \cdot w \cdot MaxReadCoverage)$, since $|I|$ cannot exceed number of *N*'s in the data. The algorithm takes approximately 10 minutes to impute around 4,000 missing values on ~27,800 reads from HCV dataset (data description is available in the Section 2.6.2).

2.4.1.4 Read Graph

We begin with the definition of the read graph, introduced in [54] and independently in [22], and describe the adjustments in read graph construction and edge weights to account for sequencing errors as well as the high mutation rate between quasispecies.

A read graph *G = (V, E)* is a directed graph with vertices corresponding to reads aligned onto the consensus sequence. For a read *u*, we denote by *b(u)*, respectively *e(u)*, the genomic coordinate at which the first, respectively the last, base of *u* gets aligned. A directed edge *(u, v)* connects read *u* to read *v* if suffix of *u* overlaps with prefix of *v* and they coincide across the overlap. Two auxiliary vertices - source *s* and sink *t* - are added such that *s* has edges to all reads with zero in-degree and *t* has edges from all reads with zero out-degree. Then each $s \rightarrow t$-path

corresponds to a possible candidate quasispecies sequence. The read graph is transitively reduced, i.e., each edge $e = (u, v)$ is removed if there is a $u \rightarrow v$-path without edge e. Note that certain reads can be completely contained inside other reads. Let a ***superread*** refer to a read that is not contained in any other read and let the rest of the reads be called ***subreads***. Subreads are not used during graph construction, so any two different $s \rightarrow t$-paths represent to two different candidate sequences.

To account for sequencing errors, we adjust the construction of the read graph to allow for mismatches. We use three parameters: (1) n = #mismatches allowed between a subread and a superread, (2) m = #mismatches allowed in the overlap between two consecutive reads, and (3) t = #mismatches expected between a read and a random quasispecies.

First, we use Algorithm 3 to find all superreads that are not contained in any other read if more than n differences are required to distinguish two sequences (see Figure 2.5).

Algorithm 3 Finding superreads.

Input

1. R is the set of reads sorted in ascending order of their starting positions.
2. n is the number of mismatches allowed between a subread and a superread.

❖ S is the set of superreads sorted in ascending order of their ending positions.

$S \leftarrow \emptyset$

for each read r in R **do**

    $toAdd \leftarrow$ TRUE

    for $j = |S|$ to 1 **do**

        if the rightmost end of the superread s_j is to the left of the rightmost end of read r **then**

```
                    break
                end if
                if r contains s_j with no more than n differences then
                    S ← S - { s_j }
                end if
                if s_j contains r with no more than n differences then
                    toAdd ← FALSE
                end if
            end for
            if toAdd is TRUE then
                S ← S U { r }
            end if
        end for
        return S
```

During graph construction, we use only superreads, but, in contract to [22, 54], subreads are taken into account in the final assembly of candidate sequences and frequency estimation. Varying parameter *n* yields different levels of population diversity captured by a read graph since larger *n* allows to "cluster" not only reads produced from the same quasispecies sequences and being different just due to sequencing errors but also reads sequenced from several (negligibly) different quasispecies sequences. Thus, smaller *n* increases chances to find infrequent but medically relevant quasispecies sequences. Oh the other hand, too small *n* results in low connectivity in the read graph. In our experiments, we varied *n* over range from 0 to 6.

The worst-case runtime for Algorithm 3 is $O(|R| \cdot \bar{l} \cdot MaxReadCoverage)$, where $|R|$ is the number of reads, and $\bar{l}$ is the average read length.

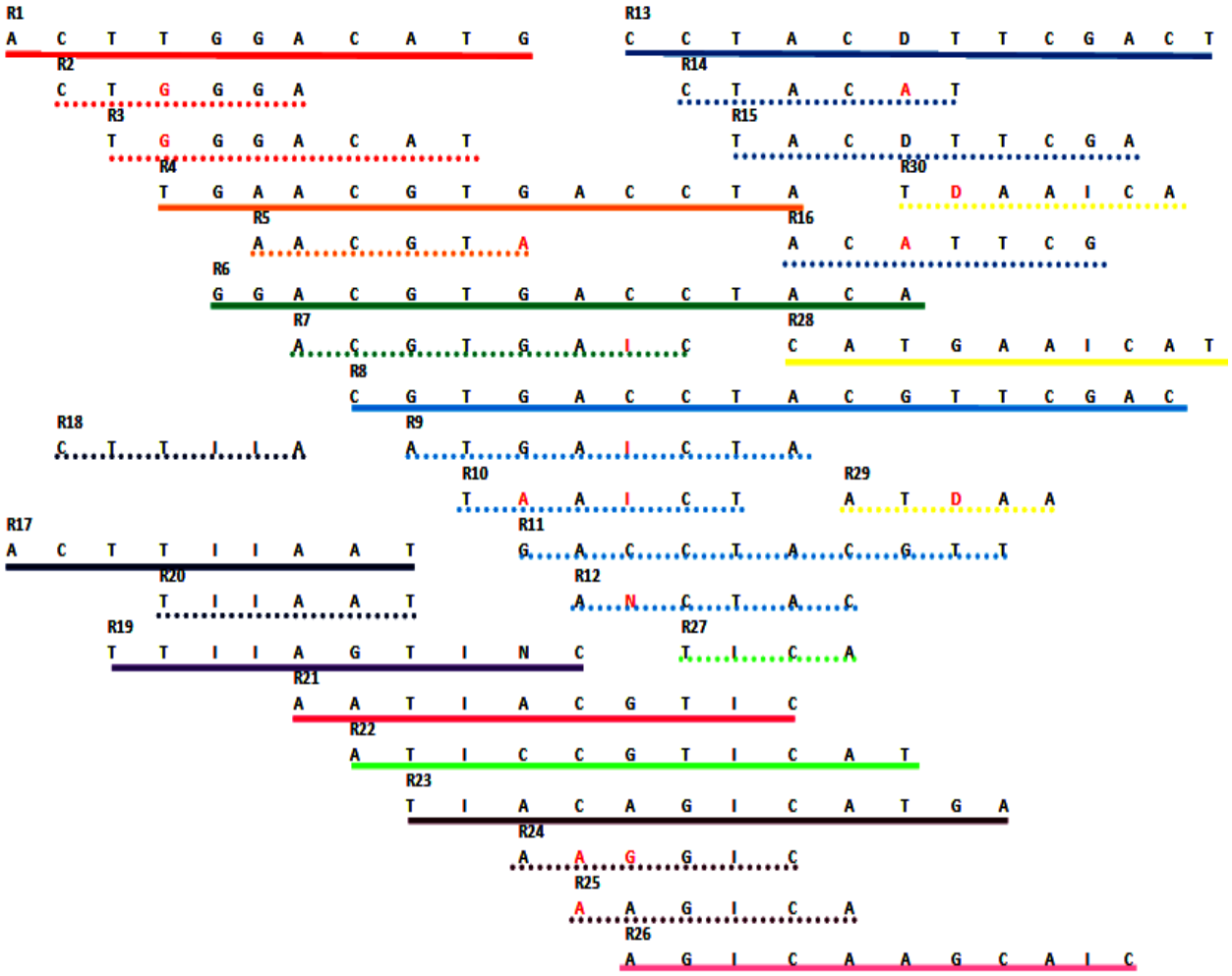

Figure 2.5 *Example of superreads (underlined by bold lines) and subreads (underlined by dashed lines) (n = 2). A superread and its subreads are underlined by the line of the same color. If a subread has a mismatch with its superread, the mismatch is colored in red.*

At the second step (see Algorithm 4), we add a directed edge between every pair of vertices if their corresponding reads partially (with at most *m* mismatches) agree in their overlap (see Figure 2.6). Parameter *m* influences the connectivity of the graph: the larger value is used, the more paths from sink to source exist in the read graph. Since we allow at most *n* mismatches between read and its superread, it is reasonable to accept up to *n* mismatches in overlap between consecutive superreads (lower bound) and up to *2n* - *3n* mismatches between their subreads.

In the worst case, Algorithm 4 is $O\left(|S| \cdot \overline{l_s} \cdot MaxSuperreadCoverage\right)$, that would not exceed $O\left(|R| \cdot \bar{l} \cdot MaxReadCoverage\right)$, where |*S*| and |*R*| respectively is the number of

superreads and reads, $\overline{l_s}$ and $\bar{l}$ are the average length of superreads and all reads, respectively. In practice, the larger *n* is, the lesser superreads are, resulting in lesser superreads coverage.

Algorithm 4 Creating edges.

Input

1. *S* is the set of sorted superreads.
2. *m* is the number of mismatches allowed in the overlap between two consecutive reads.

❖ *E* is the set of edges in the graph *G*.

$E \leftarrow \emptyset$

for each i=1 to |S|-1 **do**

  for each $j=i+1$ to |S| **do**

    if $e(s_i) \leq b(s_j)$ **then**

      break

    end if

    if s_i overlaps with s_j with no more than *m* mismatches **then**

      $E \leftarrow E \cup \{ (s_i, s_j) \}$

    end if

  end for

end for

return *E*

Following [55], we apply transitive reduction to our read graph (see Figure 2.6 (bottom)). In case no mismatches are allowed (*n=0, m=0*), read graph before transitive reduction is equivalent to transitively-reduced read graph since if there are edges *(u, v)* and *(v, w)* in the graph and reads *u* and *w* overlap then the edge *(u, w)* is implied (see Figure 2.7 (left)). In our case, edge *(u, w)*, strictly saying, does not follow from the overlap of reads *u* and *w* since it may be up

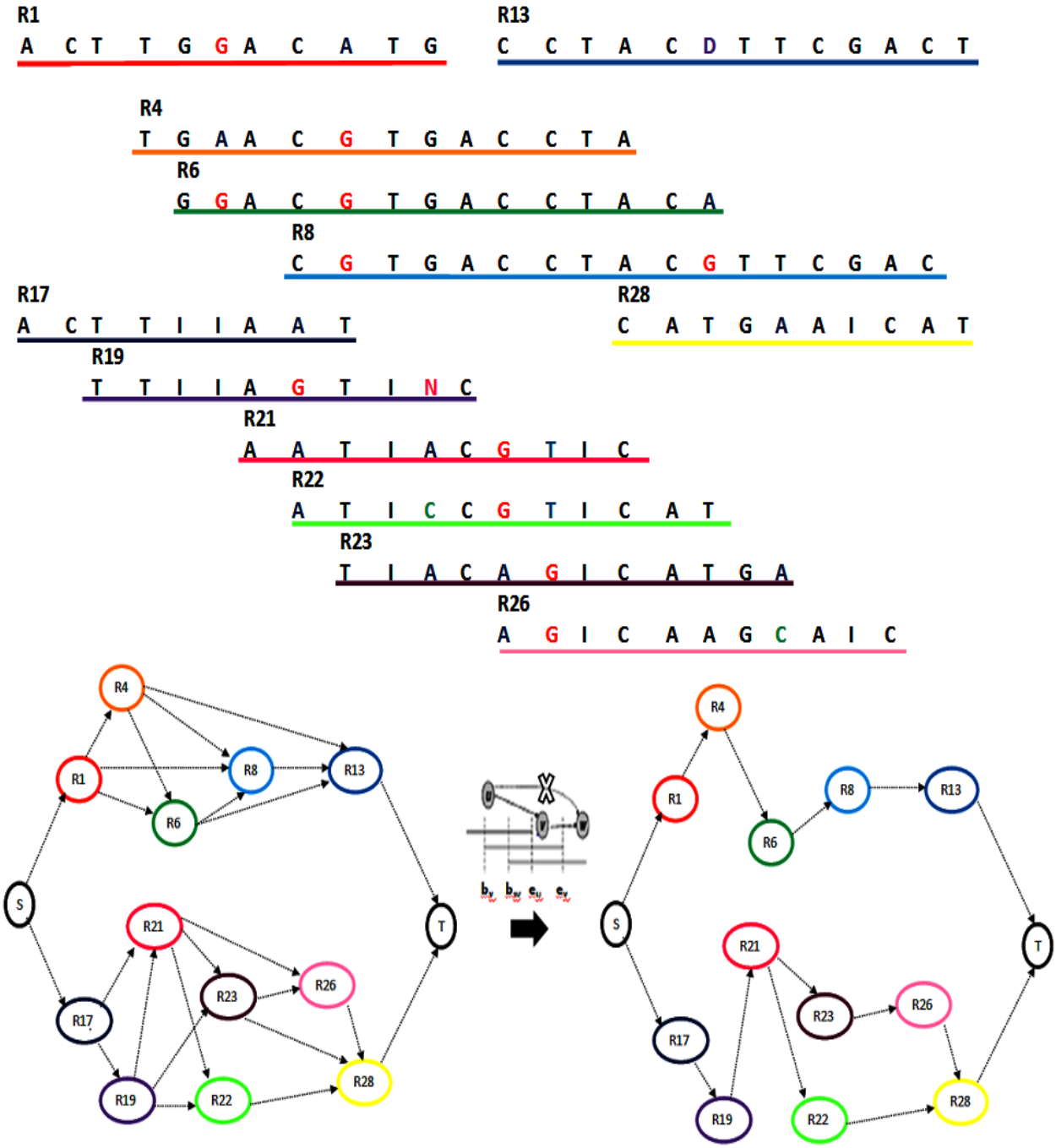

Figure 2.6 *Adding edges to a read graph. Top: to illustrate differences between superreads in overlap, we highlight those positions where several allelic values across reads are seen. For example, R1 and R4 have 2 differences in the overlap. For simplicity of presentation, differences are marked across reads R1-R13 and R17-R28, separately. For each allelic value, we use different color: red (G), dark blue (A), green (C), blue (T), pink (N) and purple (D). Bottom: read graph for n=2, m=2 before transitive reduction (at the left) and transitively-reduced read graph (at the right).*

to *2m* mismatches between their sequences in the overlap, so edge *(u, w)* does not exist if only *m* mismatches are allowed. Thus, transitive reduction introduces some fake sequences in the haplotype search space (see Figure 2.7 (right)). These sequences might contain both reads *u* and *w* whereas, in reality, the reads come from different quasispecies sequences. Estimating their abundances via EM-based algorithm (see Section 2.4.2) successfully eliminates such sequences out the final list of candidate assemblies. In contrast to the case without mismatches, haplotype sequence may be represented by several $s \rightarrow t$-paths.

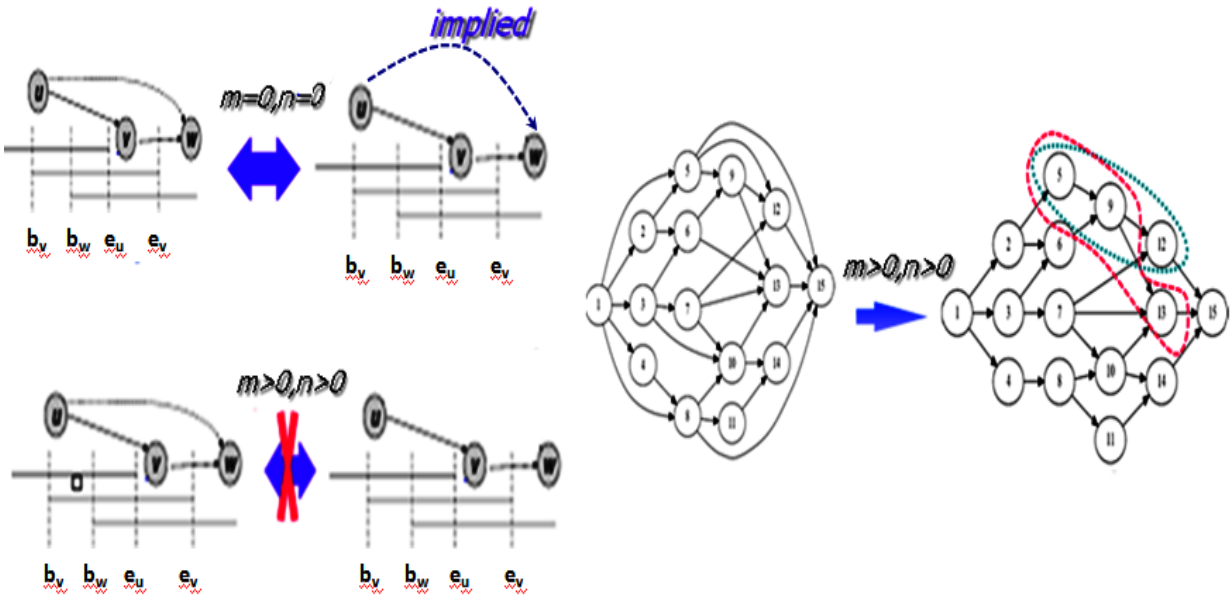

Figure 2.7 *Transitive reduction in the read graph. Left: if no mismatches is allowed (n=0, m=0) and edges (u, v) and (v, w) exist, then the edge (u, w) should exist if u and w overlap. However, if mismatches are allowed (n>0, m>0), the edge (u, w) cannot be implied. Right: if mismatches are allowed (n>0, m>0), transitive reduction introduces false positives in search space. For example, if there is an overlap between vertices 5 and 13, any path with subpath 5→9→13 represents incorrect sequence. The correct subpath is 5→9→12.*

ViSpA uses Algorithm 3 [55] to perform transitive reduction. Although the algorithm efficiently performs transitive reduction, it is one of the most time-consuming parts of ViSpA: the worst-case runtime is bounded by $O(V \cdot E^2)$ (see Appendix B).

Since the number of different *s*→*t*-paths is exponential, we wish to generate a set of paths that have high probability to correspond to real quasispecies sequences. In order to estimate path probabilities, we independently estimate for each edge *e* the probability *p(e)* that it connects two reads from the same quasispecies, and then multiply estimated probabilities for all edges on the path. Under the assumption of independence between edges, if we assign to each edge *e* a cost equal to $-\log(p(e)) = log\left(\frac{1}{p(e)}\right)$, then the minimum-cost *s* →*t*-path will have the maximum probability to represent a quasispecies sequence.

Previously [54], we estimated the probability that two reads *u* and *v*, connected by the edge *(u, v)*, belong to the same quasispecies as

$$p_{\Delta} \approx exp\left(\frac{\Delta N}{Lq}\right) = \Theta(e^{-\Delta}) \tag{2.2}$$

where $\Delta = b(v) - b(u)$ is the *overhang* (shift) between reads *u* and *v*, *N* is the number of reads, *q* is the number of quasispecies in a viral population and *L* is the number of starting positions. Thus, in this case the cost of an edge with overhang Δ can be approximated by $\Delta \propto log\left({}^{1}/_{p_{\Delta}}\right)$.

To account for sequencing errors, we re-estimate the cost of an edge as follows. First, we consider a simple model of uniformly distributed quasispecies. After a quasispecies sequence *Q* is randomly chosen, a beginning position is generated from the uniform distribution and read *r* is produced from quasispecies *Q* with *j* sequencing errors. Let *A* be an event that two reads *u*, *v* from the same quasispecies *Q* are connected with an edge *(u, v)* in the transitively reduced graph, and *j* mismatches between *u* and *v* occur in the overlap due to the sequencing errors. In the other words, the event *A* consists of the following independent events: (1) read *u* exists, starting at the position *b(u)*, (2) read *v* exists, starting at the position $b(v) = b(u) + \Delta$, (3) no read *w* from the

same quasispecies q satisfies $b(u) < b(w) < b(v)$, and (4) there are j sequencing errors in the overlap o=$e(u) - b(u)$ of reads u, v. Given N reads, originated from q quasispecies of a length L, the probability that the read u starts at a position $b(u)$ is $\frac{N}{Lq}$. The probability of the event $\Delta > k$ is the probability that there is no read from quasispecies Q, starting at any position in $\{b(u) + 1, b(u) + 2, \dots, b(u) + k\}$, that is

$$p_k = \left(1 - \frac{N}{Lq}\right)^k \approx exp\left(\frac{-kN}{Lq}\right).$$

Then

$$Pr(\Delta = k) = \left(\frac{N}{Lq}\right)^2 p_{k-1}.$$

Next we calculate probability of having j sequencing errors in overlap o, which is

$$\Pr(j) = \binom{o}{j} \varepsilon^j (1 - \varepsilon)^{o-j},$$

where ε is sequencing error rate. Finally, the probability of event A is a product of probabilities $Pr(\Delta = k)$ and $\Pr(j)$, or

$$\Pr(A) = \left(\frac{N}{Lq}\right)^2 exp\left(\frac{-\Delta N}{Lq}\right)\binom{o}{j} \varepsilon^j (1 - \varepsilon)^{o-j} \approx exp(-\Delta)\binom{o}{j} \varepsilon^j (1 - \varepsilon)^{o-j}.$$

As in the case of error-free reads, defining the edge costs as

$$\Delta\log\left(\binom{o}{j}^{-1} \varepsilon^{-j} (1 - \varepsilon)^{j-o}\right) \propto log\left({}^{1}/_{\Pr(A)}\right) \quad (2.3)$$

ensures that $s \rightarrow t$-paths with low cost correspond to most likely quasispecies sequences.

In practice, reads starting positions indeed tend to follow uniform distribution with small amount of outliers. However, if it is not a case for a particular shotgun data, cost formula (2.3) can be adjusted by taking into account the number of reads, starting at any position in $\{b(u), b(u)+1,\ b(u)+2, \ldots, b(v)\}$. Since ViSpA constructed assembly by building a path in the graph when, at each step, tool chooses among edges, spanning almost the same positions, such adjustment is non essential for ViSpA assembling.

The assumption of uniformly distributed quasispecies is most likely violated by real viral populations. Since there is no reliable way to approximate their unknown distribution, we will treat all identical copies of the same quasispecies sequence as different entities during assembling and estimate their frequencies via maximum likelihood.

Experimental results on 454 pyrosequencing shotgun reads from HCV quasispecies show that if we do not take sequencing errors into the account, that is, cost of edges is defined by (2.2), ViSpA's final assemblies correspond to non viable proteins, with many stop codons in the coding region. However, if cost is defined by formula (2.3), stop codons are disappeared in almost all cases.

In graph with E edges, cost can be assigned in $O(E \cdot \bar{o})$, where $\bar{o}$ is the length of the average overlap of the reads among all edges in E.

2.4.1.5 Candidate Sequences

Ideally, one can find candidate sequences corresponding to all $s \rightarrow t$-paths and then estimate their frequencies using maximum likelihood. Unfortunately, the number of such paths is exponential in the number of superreads. So we compute for each vertex in the read graph the

minimum cost $s \rightarrow t$-path, passing through it, to generate a set of high-probability (low-cost) paths that is rich enough to explain observed reads. Finding these paths is computationally fast. Indeed, we only need to compute in read graph G two shortest-path trees, one that is outgoing from source *s* and the other that is incoming into sink *t*; and the shortest $s \rightarrow t$-path, passing through vertex *v*, is the concatenation of the shortest $s \rightarrow v$- and $v \rightarrow t$-paths.

Preliminary simulation experiments show that better candidate sets are generated when edge costs *c* defined by (2.3) is replaced by e^c. In fact, if we use even faster dependency on *c*, we obtain better candidate sets. The fastest growing cost effectively changes the shortest path into so called ***max-bandwidth paths***, i.e., paths that minimize maximum edge cost for the entire path and for each subpath. So, ViSpA generates max-bandwidth path per each vertex. As shown in literature [65], Dijskstra's algorithm solves the single-source max-bandwidth-paths problem if we (1) initialize key d_s for the source of a tree to inifinity, (2) set keys $d_v$for all other vertices into 0, and, (3) at each step, relax those edges that satisfy the condition $d_v < \min\{d_e, cost(e)\}$. Thus, it takes order of $O(VlogV + E)$ running time to obtain set of candidate paths.

When no mismatches are allowed in the construction of the read graph, finding the candidate sequence corresponding to a $s \rightarrow t$-path is trivial since, by definition, adjacent superreads coincide across their overlap. When mismatches are allowed, we first assemble a consensus sequence from superreads used by the $s \rightarrow t$-path (see Algorithm 5). At each position, the initial candidate sequence has either allele value or placeholder (*A, C, T, G, D, I*), which is present in more than 70% of path' superreads, covering the position, or *N*, otherwise. Since several candidate paths can result in the same initial candidate sequence, we remove duplicates; but in this case, *N* matches only *N*. It could be not the best choice, especially, when the coverage

with superreads is low[1]. Hence, we replace each initial candidate sequence with a weighted consensus sequence obtained on both superreads and subreads, as described below.

For each read *r*, we compute the probability that it belongs to a particular initial sequence *s* as:

$$\mathrm{p}(s,r) = \binom{l}{k}\left(1-\frac{t}{L}\right)^{l-k}\left(\frac{t}{L}\right)^{k}, \tag{2.4}$$

where *l* and *L* denote the lengths of the read *r* and initial candidate sequence *s*, respectively, *k* is the number of mismatches between the read *r* and the initial candidate sequence *s*, and $\frac{t}{L}$ is the expected mutation rate in a viral population. Then final candidate sequence is computed as the weighted consensus over all reads, where the read weight is its probability to belong to the sequence if quasispecies sequences mutate with rate $\frac{t}{L}$. Note that, unlike the case without mismatches, the same candidate sequence can be obtained from different candidate *s*→*t*-paths so we remove duplicates, at the end of the step. The repetition of the same candidate sequence serves as evidence that this sequence is from viral population rather than assembled by chance.

Algorithm 5 Assembling final candidate sequences from the set of candidate paths.

Input

1. *S* is the set of sorted superreads.
2. *R* is the set of sorted subreads.
3. *P* is the set of candidate paths
4. *t/L* is the expected mutation rate in a viral population.

❖ *I* is the set of initial candidate sequences

[1] High coverage of the consensus does not imply high coverage of certain quasispecies sequences.

❖ *C* is the set of final candidate sequences

❖ *profile[L+1]* stores counts for each position:

- how many reads (superreads), covering the position, support *A, C, T, G, D, I, N*.

C ← ∅

for each path *p* in *P* **do**

set to 0 all counts in *profile[]* for all positions

seq ← empty string

for each superread *r* in path *p* **do**

update appropriate counts at *profile[]* for each position covered by *r*

end for

for position *i*=1 to *L* **do**

max←maximum count among counts for *A, C, T, G, D, I, N* in *profile[i]*

allele←allele or placeholder for which count is equaled to *max* in *profile[i]*

total ←sum of all counts in *profile[i]*

if *max/total*<0.7 **then**

seg←*seg*+*N*

else

seg←*seg*+*allele*

end if

end for

if *seq* is not in *I* **then**

I←*I U*{*seq*}

end if

end for

for each sequence *s* in *I* **do**

set to 0 all counts in *profile[]* for all positions

seq ← empty string

for each read *r* in *R U S* **do**

calculate *p(s, r)* by formula (2.4)

for each position covered by *r*, update appropriate counts at *profile[]* by adding *p(s,r)*

end for

for position *i*=1 to *L* **do**

allele←allele or placeholder for which count is maximal in *profile[i]*

seg←*seg*+*allele*

end for

if *seq* is not in C **then**

C←*C U*{*seq*}

end if

end for

return *C*

In the worst case, Algorithm 5 is $O\left(cnst \cdot |S| \cdot \overline{l_s} + |I| \cdot |R \cup S| \cdot \bar{l}\right) = O\left(|I| \cdot |R \cup S| \cdot \bar{\bar{l}}\right)$, which cannot exceed $O\left(|P| \cdot |R \cup S| \cdot \bar{\bar{l}}\right)$, where $|S|$ and $|R \cup S|$ respectively is the number of superreads and reads, $\overline{l_s}$ and $\bar{l}$ are the average length of superreads and all reads, respectively, *cnts* is some constant, and $|P|$ is the number of paths in the candidate set and can be as high as number of superreads if connectivity is low in the graph. Obviously, assembling final candidate sequenced from candidate paths is a computational bottleneck for ViSpA (see Appendix B).

Large number of candidate sequences or low diversity among them may result in less accurate estimate of their frequencies. Optionally, we can reduce amount of candidate sequences by clustering and replacing them with the set of consensuses obtained in each cluster. Although

clustering lowers chances to find rare sequences, it may be useful, for example, if viral sample contains several populations from different patients, and we would like to partition assembled quasispecies among them. If the clustering option is used on viral population from one patient, we first estimate minimum number *h* of quasispecies sequences, required to explain the observed reads, by network flow approach [54, 55] and then cluster all candidate sequences into *h* clusters based on the Hamming distance.

2.4.2 Frequency Estimation for Quasispecies Sequences

Once a set of candidate sequences is obtained, their maximum-likelihood frequencies are calculated by EM-based algorithm.

First, we assume that reads *R* with observed frequencies $\{o_r\}_{r=1}^{|R|}$ where generated from a quasispecies population Q as follows. First, a quasispecies sequence $q \in Q$ is randomly chosen accordingly to its unknown frequency f_q. A read starting position is generated from the uniform distribution and then a read r is produced from quasispecies q with j sequencing errors. The probability of this event is calculated as

$$h_{q,r} = \binom{l}{j}(1-\varepsilon)^{l-j}\varepsilon^{j},$$

where l is the read length, and ε is the sequencing error rate. Thus, the probability of observing a read r under this model is $\Pr(r) = \sum_{q\in Q} f_{\mathrm{q}}\, \mathrm{h_{q,r}}$. Then quasispecies frequencies $\{f_q\}_{q=1}^{|Q|}$ are estimated by maximizing the log-likelihood function:

$$\ell(\mathrm{f_1}, \ldots \mathrm{f_{|Q|}}) = \sum_{\mathrm{r}\in\mathrm{R}} \mathrm{o_r} \log \Pr\,(\mathrm{r})$$

using an EM algorithm [66].

After initializing frequencies $\{f_q\}_{q\in Q}$ at random, the algorithm repeatedly performs the next two steps until convergence:

- **E-step:** For each pair q, r, compute the expected value $p_{q,r}$ that read r comes from candidate sequence q under the assumption that frequencies $\{f_q\}_{q\in Q}$ are correct by the following formula:

$$p_{q,r} = \frac{f_q \cdot h_{q,r}}{\sum_{q':(q',r)\in E} f_{q'} \cdot h_{q',r}}.$$

- **M-step:** For each $q \in Q$, update value of f_q to the portion of reads being originated by the candidate sequence q among all observed reads in the sample, that is:

$$f_q = \frac{\sum_{r:(q,r)\in E} p_{q,r} \cdot o_r}{\sum_{r\in R} o_r}.$$

Currently convergence of EM algorithm is determined at the tolerance level 0.005.

2.5 Validation

The validation of our method consists of assembling validation and validation of maximum likelihood frequencies obtained by EM algorithm.

2.5.1 Quasispecies Assembling Validation

Validation of assemblies obtained on the synthetic reads (see 2.6.1) is straightforward since original quasispecies sequences are available. We say that the quasispecies sequence is captured if one of the candidate sequences exactly matches it. If we compare sequences of negligibly different lengths, we trim longer sequence to the shorter length before comparison. The quality of assembling is measured by portion of the real quasispecies sequences being

captured by candidate sequences, or ***sensitivity*** = TP/(TP+FN), and its portion among candidate sequences in cross-validation tests, that is, positive predictive value ***PPV*** = TP/(TP+FP).

Clearly, we cannot use the same validation for quasispecies assembled on 454 pyrosequencing shotgun reads (see 2.6.2). In this case, the best validation is a biological confirmation in lab experiments. Due to the cost of lab experiments, it may not be a viable solution. Hence, we suggest internal validation via bootstrapping tests to measure reproducibility of the top ten most frequent quasispecies.

In bootstrapping test, a subsample of the original data (sample) is analyzed. Each subsample is created by randomly choosing reads from the original sample. Since reads are chosen number of times, equaled to the size of the original sample, and repetitions are allowed, a number of distinct reads in the subsample is 2/3 of the reads in the sample.

On each subsample, we run ViSpA and find the best unique match between top ten most frequent candidate sequences assembled on the sample and top ten most frequent assemblies obtained on the current subsample. Both sets of sequences are sorted in decreasing order of their EM frequencies and traversed, starting from the most frequent assemblies. Then for each assemblies produced on the sample, we find the closest unmatched candidate sequence on the subsample. Two distinct candidate sequences assembled on the sample would not match the same assemblies obtained on the subsample. The higher EM frequency is associated with the candidate sequences, the closer sequence would be matched to it. For each found match, we report the number of mismatches between the sequences.

After sufficient number of bootstrapping tests, we record reproducibility of each out of top assemblies obtained on the sample by counting percentage of bootstrapping runs when there is a match(exact or with at most *k* mismatches) among 10 most frequent sequenced found on

reduced data. Higher percentage corresponds to the more robust (reproducible) assembled sequence. Reproducibility serves as a confidence that it is viral quasispecies sequence from population rather than sequence assembled by chance.

Since removing of 1/3 of reads can significantly reduce underlying genomic diversity, we mostly run tests on 10%-reduced data sets.

2.5.2 Validation of Frequency Distribution

Following [22], we measure the prediction quality of frequency distribution with Kullback-Leibler divergence, or relative entropy in the case of the simulated data. If two probability distributions are given then relative entropy measures the "distance" between them, or, in the other words, the quality of approximation of one probability distribution by the other distribution. Formally, the relative entropy between true distribution P and approximation distribution Q is given by the formula:

$$D_{KL}(P||Q) = \sum_{i \in P} P(i) log \frac{P(i)}{Q(i)},$$

where summation is over all reconstructed original sequences $I = \{i|P(i) > 0, Q(i) > 0\}$, that is, over all original sequences that have a match (exact or with at most k mismatches) among assembled sequences.

2.6 Data Sets

2.6.1 Reads Simulated from Known HCV Quasispecies

In order to perform cross-validation tests on the assembly method, we simulate reads data from 1,739-bp long fragment of E1E2 region of 44 HCV sequences [26] with sequence frequencies followed particular distribution. In our simulation experiments, we use uniform distribution, geometric distribution (when i^{th} sequence is constant factor more frequent than the

($i+1$)st sequence), and skewed distribution (when one sequence has very high frequency whereas the others follow a uniform distribution). For each of the distributions, we create sample quasispecies populations with different number of randomly selected above-mentioned quasispecies sequences.

We first simulate error-free reads, e.g., reads without indels with respect to the reference sequence and without sequencing errors. The length of a read follows normal distribution with a particular mean value and variance 400 and a starting position follows the uniform distribution. This simplified model of reads generation has two parameters: number of the reads that varies from 20,000 up to 100,000, and the average read length that varies from 200 bases up to 600 bases.

In the second set of experiments, we simulate 454 pyrosequencing shotgun reads for 10 random quasispecies sequences (under geometric distribution) out of 44 HCV sequences [26] using FlowSim [6]. We generated 39,131 reads with length varying from 50 bases up to 550 bases and average read length equaled to 322 bases (see Figure 2.8 (left)). Each position (except

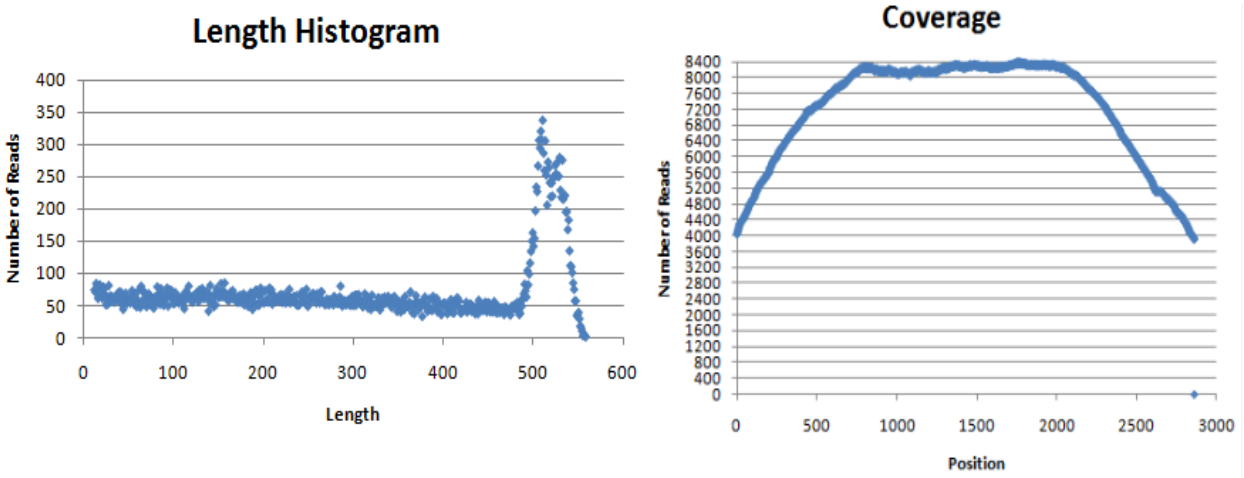

Figure 2.8 *Length histogram and position coverage for reads generated by FlowSim. Left: number of aligned reads with given length. Right: number of aligned reads covering extended reference positions. Each position is covered by at least 4000 reads, except the position at the very end.*

the end) is covered by at least 4000 reads (see Figure 2.8 (right)).

Additional Read Statistics: 99.96% of aligned reads has at least one indel with respect to the reference; 99.97% of deletions and 99.6% of insertions are 1bp-long. Only 1.1% of the aligned reads has unknown value(s).

2.6.2 454 Pyrosequencing Reads from HCV Sample

The data set Data1 has been received from Peter Balfe, HCV Research Group in Institute of Biomedical Research at University of Birmingham. Data1 contains 30,927 reads obtained from the 5.2kb-long fragment of HCV-1a strain (which is more than a half of the entire HCV genome). The host is an intravenous drug user being infected for less than 3 months, so the expected mutation rate is between 1.75% and 8%. 27,764 reads were aligned versus reference by SEGEMEHL software [49].

The aligned read length average is 292 bases but it significantly varies (see Figure 2.9) as

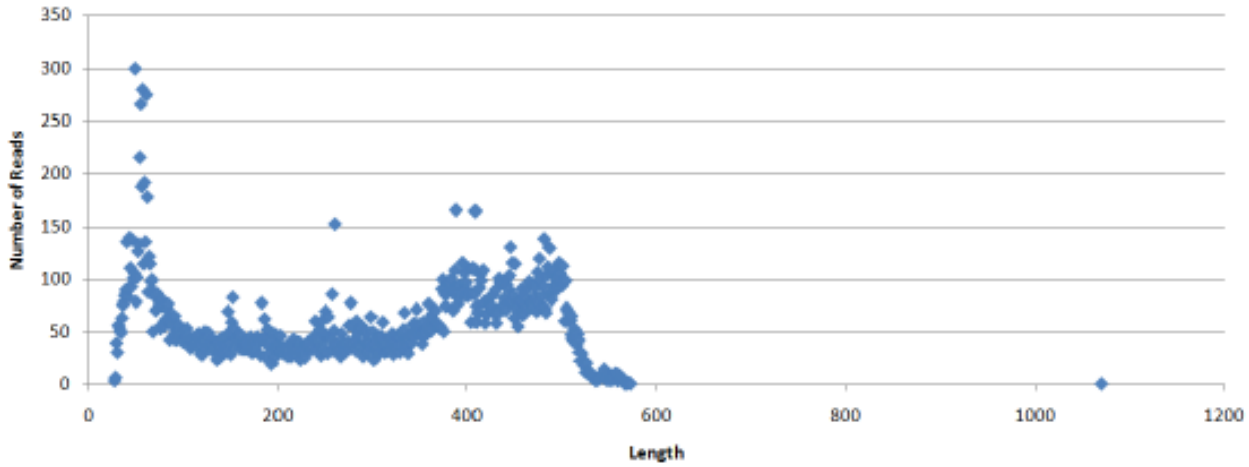

Figure 2.9 *Number of aligned reads with given length (HCV data). There is a single read 1050bp long, major peak at 450bp long and 50bp long.*

well as the depth of position coverage (see Figure 2.10). The depth of reads coverage variability is due to a strong bias in the sequence start points, reflecting the secondary structure of the

template DNA or RNA used to generate the initial PCR products. As a result, shorter reads are produced by GC-rich sequences.

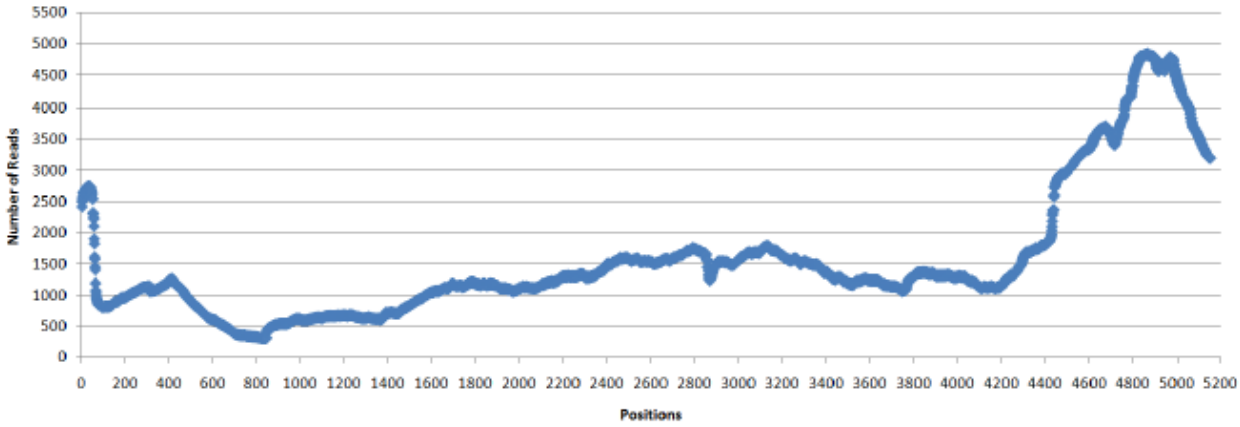

Figure 2.10 *Number of aligned reads covering extended reference positions (HCV data). The global minimum is 800th nucleotide covered by 310 reads.*

Additional Read Statistics. 72% of the aligned reads has at least one deletion with respect to the reference: 98% of deletions are 1 base long, 1.5% has length 2, and the rest 0.5% has length 3. 77% of the aligned reads has at least one insertion: 86% of insertions have length equaled to 1, and 9.8% have length equaled to 3. Only 7% of aligned reads has at least one unknown value. We estimate that the insertion error rate is at least 0.025% under assumption that only once encountered insertions is caused by sequencing errors.

2.6.3 454 Pyrosequencing Reads from HIV Sample [58]

The HIV dataset contains 63,019 reads from mixture of 10 different 1.5kb-long region of HIV-1 clonal fragments, including *pol* protease and part of the *pol* reverse transcriptase. 55,611 reads were aligned versus HXB2 reference [2] by SEGEMEHL [49]. The aligned reads length varies from 35 bases up to 584 bases with average read length 345 bases (see Figure 2.11). In contrast to [58], we do not filter out low-quality reads, that is, reads with a *phred* score below 10 at least at one sequenced position.

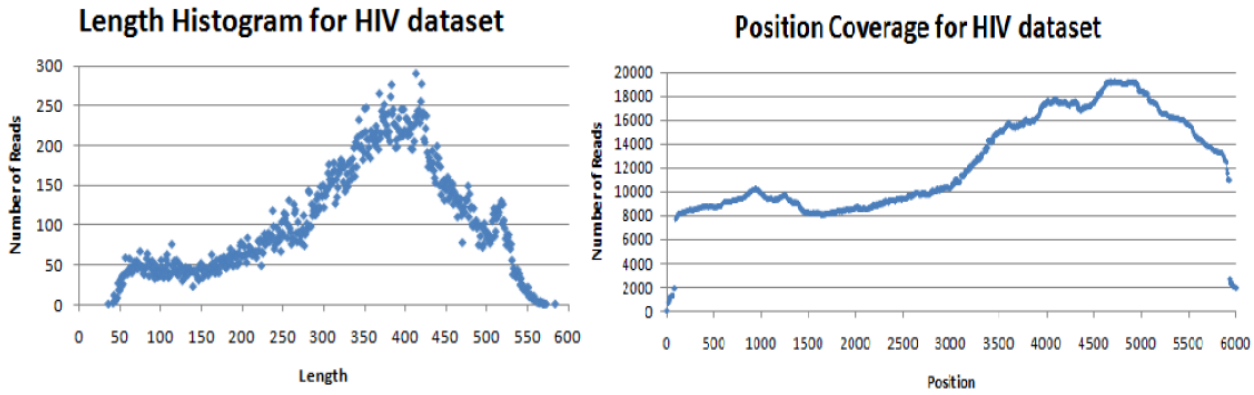

Figure 2.11 *Length histogram and position coverage for HIV data. Left: number of aligned reads with given length. There is a major peak around 400bp long. Right: number of aligned reads covering extended reference positions. Each position is covered by at least 8000 reads, except positions at the very beginning and at the very end.*

Additional Read Statistics. 87% of the aligned reads has at least one deletion with respect to the reference: 99.97% of deletions are 1bp long. 99% of aligned reads has at least one insertion: 85% of insertions have length equaled to 1, 10% have length equaled to 2, and 3.5% have length equaled to 3. 11.6% of the aligned reads has at least one unknown value.

2.7 Experimental Validation

Below we discuss results of experiments on the simulated and real pyrosequencing shotgun reads.

2.7.1 Simulated Data

In our experimental validations, we compare the proposed algorithm ViSpA with the state-of-the-art ShoRAH [57] as well as with ViSpA on ShoRAH-corrected reads (ShoRAHreads+ViSpA).

On the error-free reads, ViSpA[2] significantly outperforms ShoRAH. Both sensitivity and PPV are analyzed as functions of the number of quasispecies in underlying sample population

[2] All experimental results for ViSpA on error-free reads are available in appendix C.

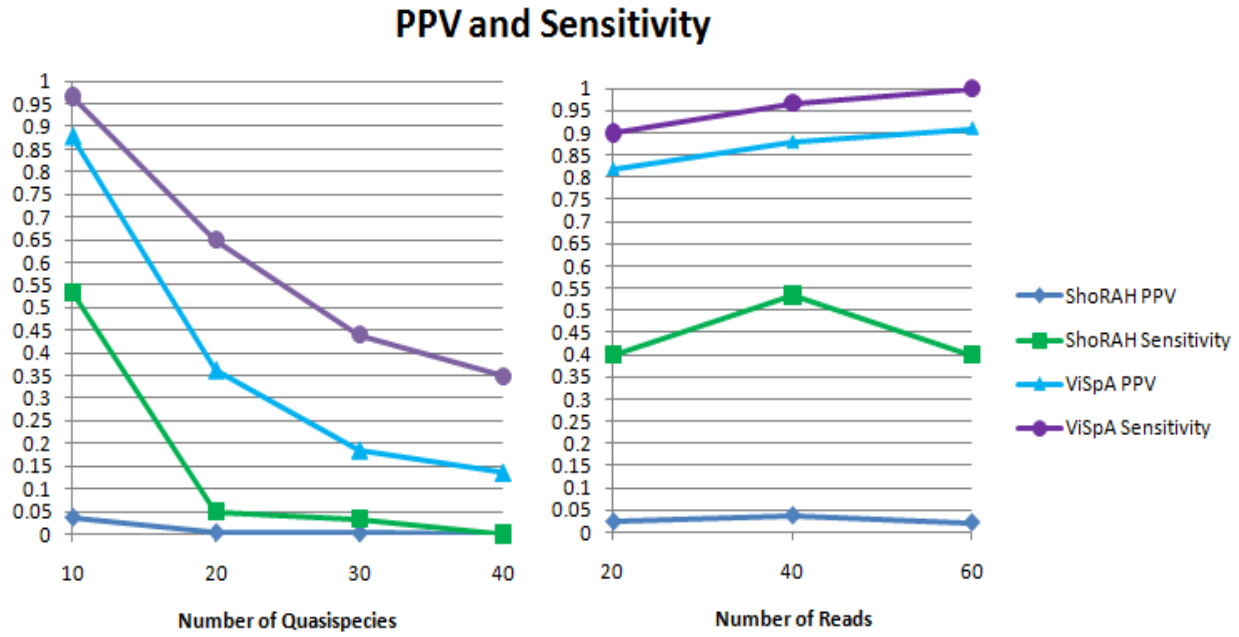

Figure 2.12 *Left: PPV and sensitivity as a function of the number of quasispecies in the original population (40K reads with average read length 300, geometric distribution). Right: PPV and sensitivity as a function of the number of reads (10 quasispecies under geometric distribution, average read length is 300bp).*

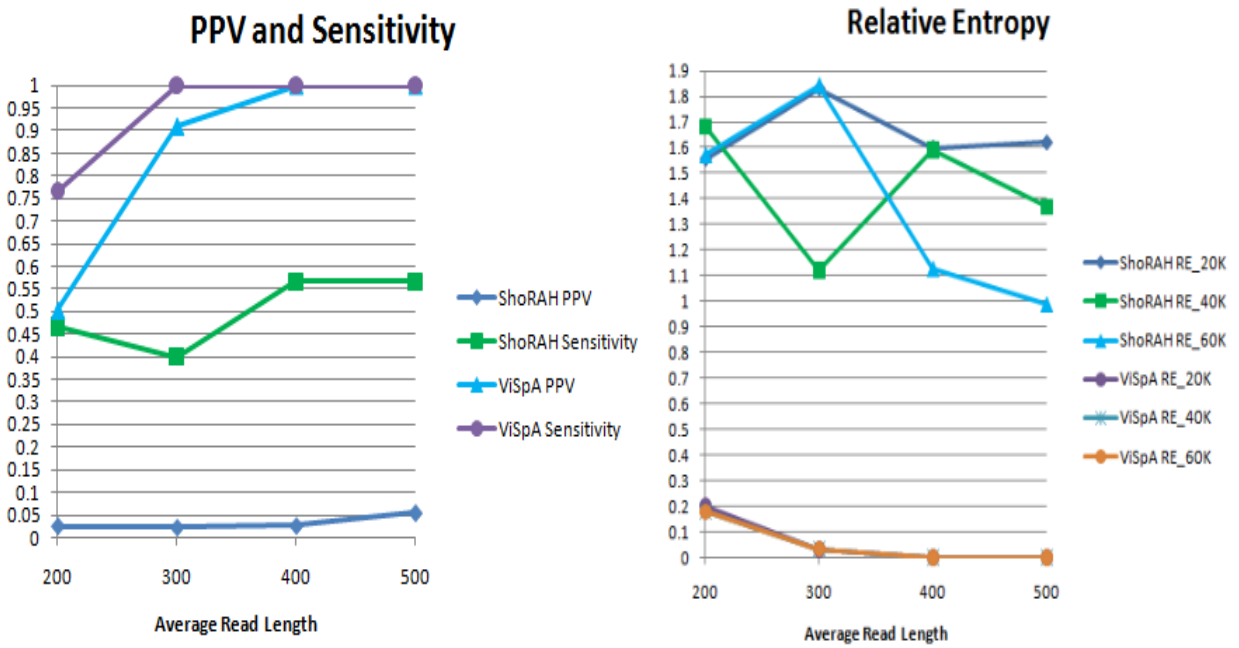

Figure 2.13 *Left: PPV and sensitivity as a function of the read average length (60K reads, 10 quasispecies under geometric distribution). Right: the relative entropy as a function of the average read length (40K reads from 10 quasispecies).*

(see Figure 2.12 (left)), as functions of the number of generated reads (see Figure 2.12 (right)) and as functions of average read length (see Figure 2.13 (left)). ViSpA can correctly assemble all sequences for 10 quasispecies if average read length is at least 300 bases and 26 sequences out of 40 quasispecies if average read length is at least 400 bases. If the average read length is smaller (for example, in range from 250bp till 299bp), the method can assemble at least 8 out of 10 sequences and 20 out of 40 sequences. ShoRAH assembles 5-6 sequences out of 10 and at most 1 out of 20 sequences. Moreover, ViSpA gains prediction power if number of reads increases whereas there is no obvious tendency for ShoRAH.

In case of ViSpA, relative entropy is decreasing with increasing of the average read length (see Figure 2.13 (right)). It is expected since sensitivity is increasing with increasing of the average read length (see Figure 2.13 (left)) and EM gives closer approximation of underlying distribution. Significantly poorer performance of ShoRAH on error-free reads can mean that ShoRAH error correction algorithm is prone to overcorrection.

However, ShoRAH has a significant advantage over ViSpA on a read data simulated by FlowSim both in prediction power and in robustness of results[3] (see Table 2.1). Indeed, ShoRAH correctly infers 3 out of 10 real quasispecies sequences whereas ViSpA reconstructs only 1 sequence. Additionally, 10 most frequent assemblies inferred by ShoRAH are more robust with repeating up to 45% of times on 10%-reduced data versus 1% of times for ViSpA's assemblies. This advantage can be explained by superior read correction in ShoRAH. If ViSpA is used on ShoRAH-corrected reads, the results drastically improve: 5 quasispecies sequences are inferred and exactly 95% of times are repeated on reduced data (see Table 2.1), confirming that ViSpA is better in assembling sequences.

[3] All experimental results for ViSpA, ShoRAH and ViSpA on ShoRAH-corrected data on FlowSim generated reads are available in appendix D.

Table 2.1 Comparison of three methods - ViSpA, ShoRAH, and ShoRAHreads+ViSpA - on the read data simulated by FlowSim. The quasispecies sequence is considered found if one of candidate sequences matches it exactly ($k = 0$) or with at most k (1 or 9) mismatches. All methods were run 100 times on 10% -reduced data. For the i-th ($i=1,\ldots,10$) most frequent sequence assembled on the whole data, we record its reproducibility, i.e., percentage of runs where there is a match (exact or with at most k mismatches) among 10 most frequent sequences found on reduced data. "Reproducibility: Max" and "Reproducibility: Average" respectively report maximum and average of those percentages.

	ShoRAH				ViSpA				ShoRAHreads+ViSpA			
	PPV	Sensitivity	Reproducibility		PPV	Sensitivity	Reproducibility		PPV	Sensitivity	Reproducibility	
			Max	Average			Max	Average			Max	Average
k=0	0.0097	0.3	0.45	0.11	0.0008	0.1	0.1	0.1	0.5	0.5	0.95	0.95
k=1	0.0129	0.4	0.6	0.32	0.0008	0.1	0.1	0.1	0.5	0.5	0.95	0.95
k=9	0.0162	0.5	0.95	0.64	0.0015	0.2	0.1	0.1	0.5	1	0.95	0.95

2.7.2 Analysis of HCV data

Below we analyze results of the viral quasispecies spectrum reconstruction from Data1. We first discuss the choice of parameters of the read graph and candidate sequence assembly from $s \rightarrow t$-paths. Then we give statistical validation for obtained 10 most frequent quasispecies sequences. Finally, we show how to identify and fix the erroneous homopolymer indels in the coding region of HCV.

2.7.2.1 Analysis of the Aligned Reads without Insertions

Initially, we remove all insertions from the aligned reads and infer quasispecies spectrum based on the read graphs constructed with various numbers n and m (numbers of mismatches allowed for superreads and overlaps corresponding to edges). We sort the estimated frequencies in descending order and count the number of sequences which cumulative frequency is 85%, 90%, and 95%. Figure 2.14 reports these numbers as a percent of the total number of candidate

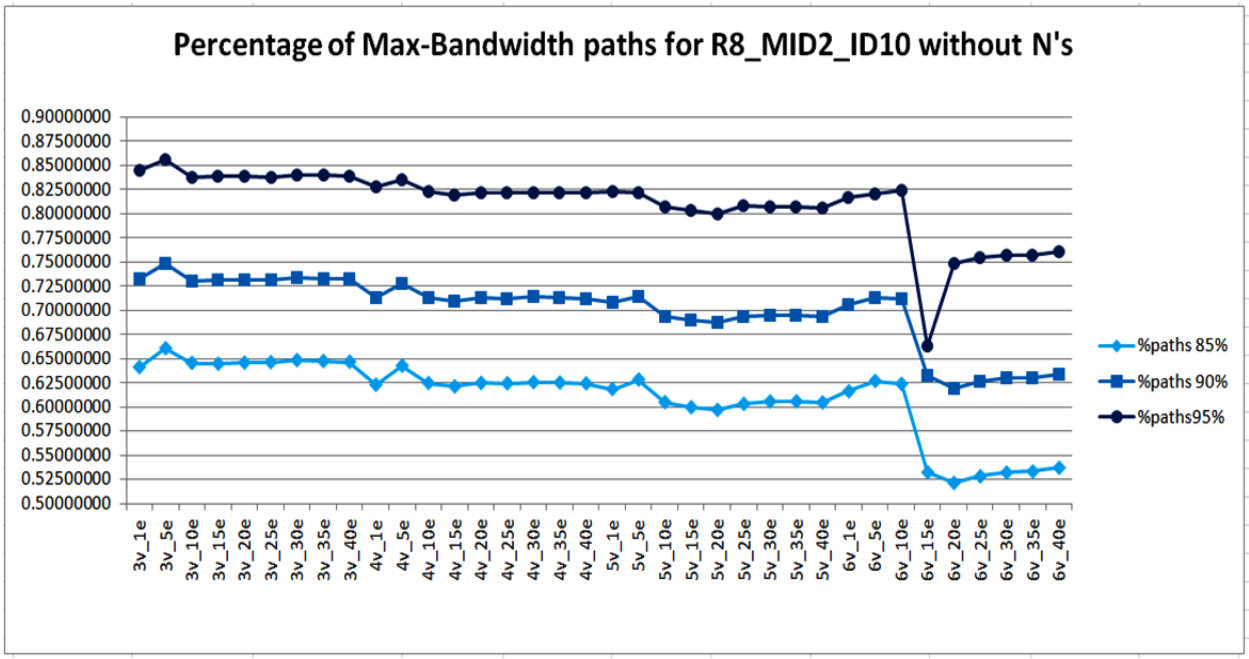

Figure 2.14 *Percentage of candidate sequences which cumulative frequencies are 85%, 90%, and 95%. The values on x-axis correspond to the number of allowed mismatches during read graph construction. n_m means that up to n mismatches are allowed in superreads and up to m mismatches are allowed in edges.*

sequences. There is an obvious drop in percentage for all three categories if we allow up to $n = 6$ mismatches to cluster reads and up to $m = 15$ mismatches to create edges. In this case, the constructed read graph has no isolated vertices.

Out of 940 candidate max-bandwidth paths per-vertex, there are only 156 initial candidate sequences, that is, sequences inferred based on superreads information. On average, the initial candidate sequences have up to 2.5% of missing values if missing values were imputed during preprocessing step or up to 2.8% of missing values if tagging method was not used. Since genotyping error is 0.1%, we do not distinguish sequences within 6 mismatches. Then there are only 58 candidate sequences.

Based on the data, we tried to estimate the 454 sequencing error. First, we counted the number i_{error} of insertions that are confirmed only by a single read out of all reads covering the position. Then optimistic estimation of the insertion error rate is i_{error}/n_{pos}= 0.000253 where n_{pos}

is the number of reads, covering this position. Being on the pessimistic side, we assume that if at most 6 insertions are among all reads, covering the position, are counted then this estimation is 0.001203. We use either pessimistic estimate or reported Roche/454 sequencing error in our EM-based algorithm as value for ε (see Section 2.4.2).

After estimating EM frequencies, we got 12 candidate sequences with frequencies at least 0.01. The most frequent candidate sequence exactly covers 26.9% of the aligned reads and 50.4% of the reads if one mismatch is allowed. In sum, these 12 candidate sequences exactly cover around 35.6% of the aligned reads and 64.5% of the reads with one mismatch. The neighbor-joining tree of these 12 candidate sequences (see Figure 2.15) reminds a neighbor-joining tree for HCV quasispecies evolution. Additionally, the most frequent candidate sequence is 99% identical to one of the

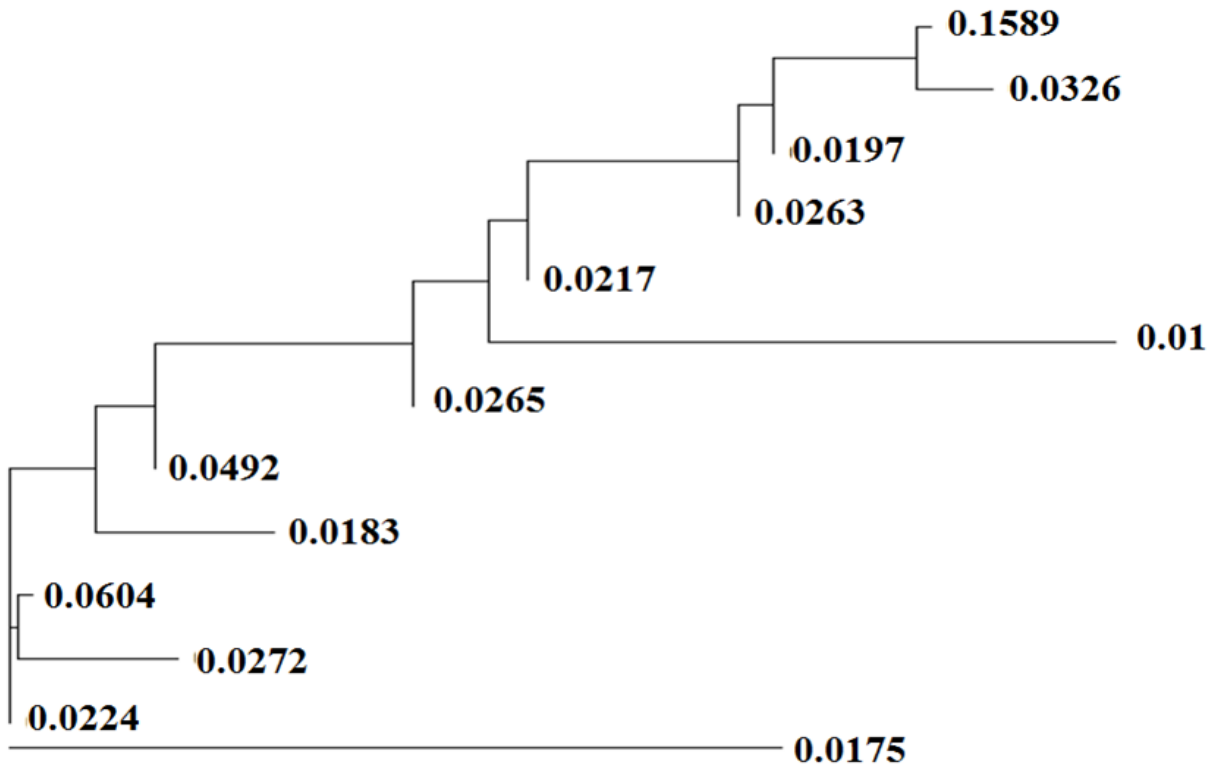

Figure 2.15 *The neighbor-joining phylogenetic tree for 12 most frequent HCV quasispecies variants on a 5,205bp-long fragment. Each variant is supplied with the estimated frequency*.

actual ORFs obtained by cloning the quasispecies. Finally, the corresponding amino-acid sequences contain starting coding sequence MSTNP and no stop codons in any of the assemblies.

Next we explore if the choice of values for *n* and *m* impacts assembly. Since sequencing error rate is 0.1%, it is reasonable to allow one mismatch to cluster reads and, as a result, two mismatches in overlap. We find 12 most frequent assembled candidate sequences and calculate the Hamming distance between each of 12 obtained candidate sequences and each of 12 candidate sequences obtained in the previous assembling (see Figure 2.16). The comparison shows that 11 most frequent candidate sequences in both cases are the same. We repeat such comparison for different values *n* and *m* ($1 \leq n \leq 6$, and $2 \leq m \leq 15$). In each case, 10-11 most frequent candidate sequences are the same among runs.

6_15\ 1_2	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	69	65	18	2	67	46	7	23	90	90
2	1	0	70	66	19	3	68	47	8	24	91	91
3	69	70	0	3	50	66	6	16	78	72	3	3
4	46	47	16	12	27	43	14	0	55	49	37	37
5	18	19	50	46	0	16	48	27	27	23	71	71
6	7	8	78	74	27	11	76	55	0	32	99	99
7	67	68	6	3	48	64	0	14	76	70	16	18
8	65	66	3	0	46	62	3	12	74	68	14	14
9	23	24	72	68	23	22	70	49	32	0	93	93
10	90	91	3	14	71	87	16	37	99	93	0	2
11	2	3	66	62	16	0	64	43	11	22	87	87
12	30	31	99	95	48	32	97	76	37	53	120	120

Figure 2.16 *Hamming distance between each of 12 most frequent candidate sequences assembled with at most 6 mismatches to cluster reads and at most 15 mismatches to create an edge (6_15) and each of 12 most frequent candidate sequences assembled with at most one mismatch to cluster reads and at most two mismatches to create an edge(1_2).*

2.7.2.2 Analysis of the Aligned Reads with Insertions

Similarly, we infer initial candidate sequences on the read graph constructed with parameters n = 6 and m = 15. To refine assembled candidate sequences, we use all reads and parameter t varying from 80 bases till 350 bases, or, in the other words, mutation rate varying from 1.75% up to 8% per sequence (which is in the range observed in [47] for acute phase of HCV). On average, final candidate sequence is within 309 bases from initial candidate sequence (see Figure 2.17).

Out of 3207 max-bandwidth paths, we obtain as much as 938 distinct sequences (t = 80) and as low as 755 sequences (t = 350) for different values of t in range [80, 350]. Since we decrease variability of the data by error correction and clustering subreads near superreads, we need to explore different mutation rates.

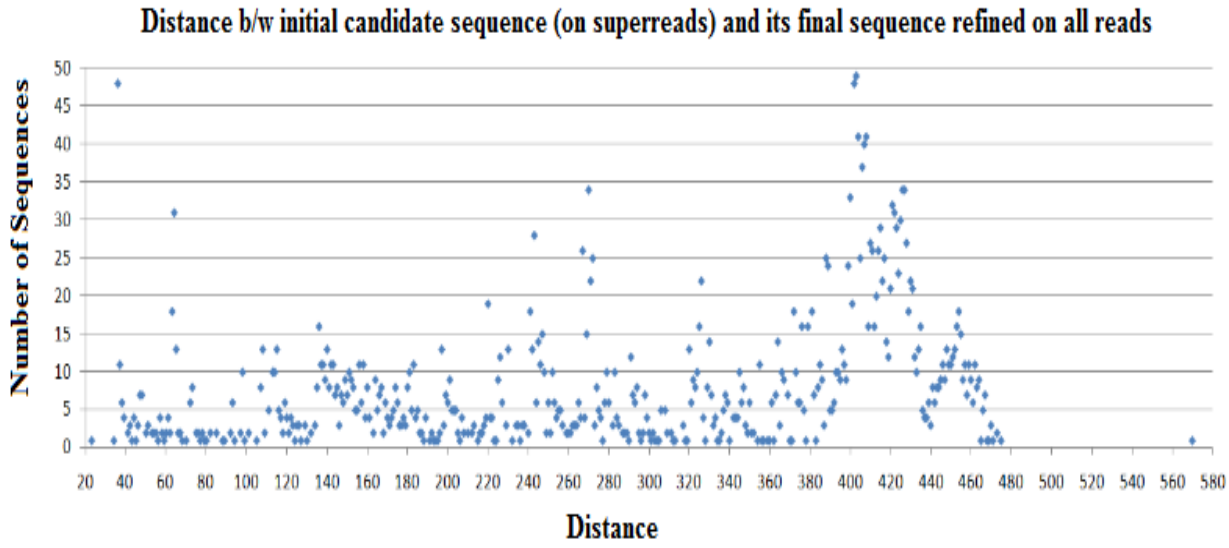

Figure 2.17 *Distance between candidate sequence assembled only on superreads and its final candidate sequence assembled on all reads. Average distance is 309 bp.*

The neighbor-joining tree for the most frequent 10 candidate sequences (see Figure 2.18) obtained by ViSpA and ShoRAH reminds a neighbor-joining tree for HCV quasispecies

evolution. Additionally, the most frequent candidate sequence found by ViSpA is 99% identical to one of the actual ORFs obtained by cloning the quasispecies.

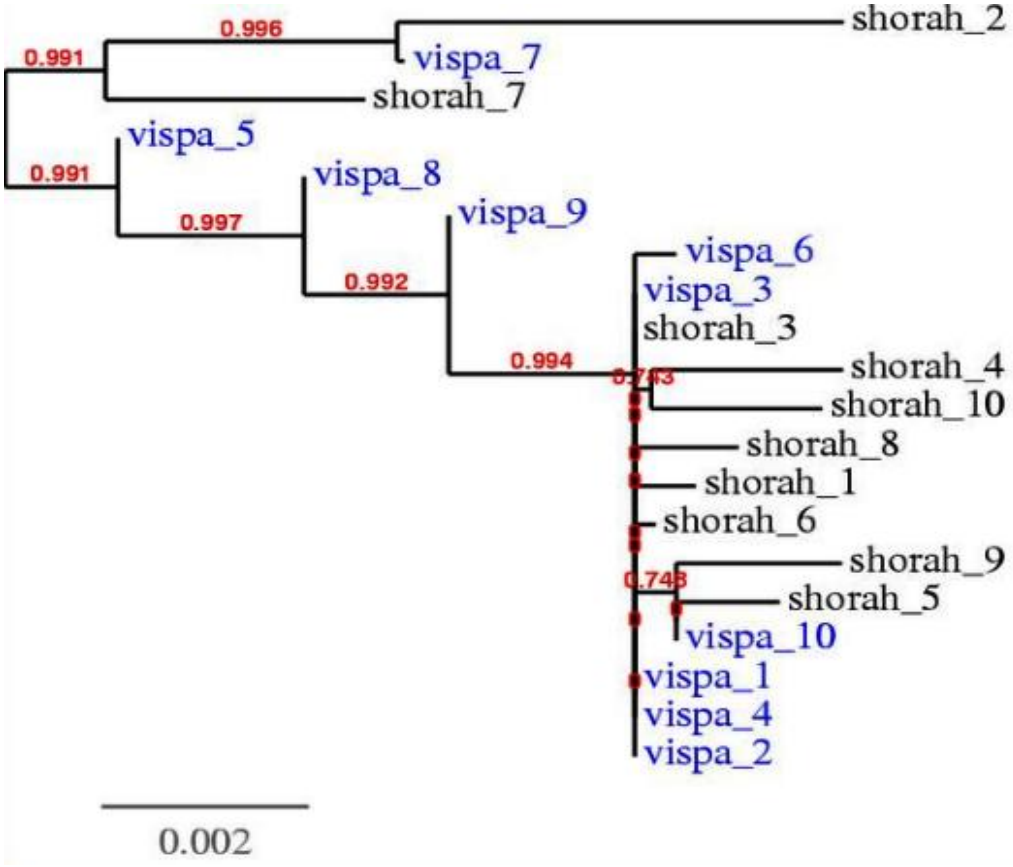

Figure 2.18 *The neighbor-joining phylogenetic tree for 10 most frequent HCV quasispecies variants on a 5,205bp-long fragment obtained by ViSpA and ShoRAH.*

Viral sequences, containing internal stop codons, are not viable since the entire HCV genome consists of a single coding region for a large polyprotein. So the number of reconstructed viable sequences can serve as an accuracy measure for quasispecies assembly. Out of 10 most frequent sequences reconstructed by ViSpA, only 3 sequences are not viable while ShoRAH is able to reconstruct only one viable sequence. This sequence has 99.94% similarity with the ViSpA's fourth most frequent assemblies. Both methods returned similar frequency estimations for this sequence: 0.017% (ShoRAH) and 0.019% (ViSpA).

The plot on Figure 2.19 shows validation results for 10 most frequent quasispecies sequences with respect to EM estimations assembled on Data1 by ShoRAH and ViSpA[4] (n = 6, m = 15, and t = 120). Repeatedly, 100 times we have deleted randomly chosen 10% of reads and

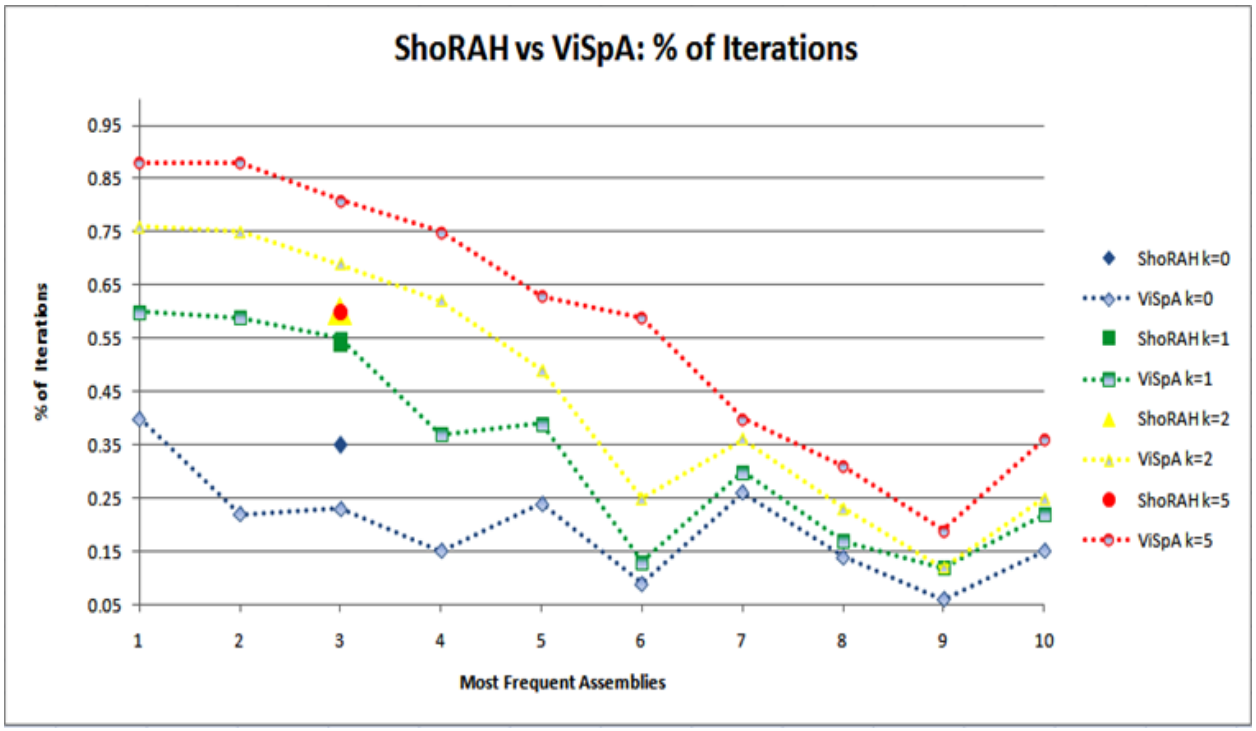

Figure 2.19 *Percentage of runs when the i-th most frequent sequences is reproduced among 10 most frequent quasispecies assembled on the 10%-reduced set of reads. The i-th point at x-axis corresponds to the i-th most frequent sequence assembled on the 100% of reads. No data are shown for the sequences that are reproduced less than 5% of runs.*

run both methods on each reduced read instance to reconstruct quasispecies spectrum. The plot reports the percentage of runs when each of 10 most frequent sequences assembled on Data1 are reproduce among the 10 most frequent quasispecies inferred on the reduced instances with no mismatches (k = 0), or with k = 1, 2, 5 mismatches. For example, for $k = 0$ ShoRAH repeatedly (35% of times) reconstructs only the 3rd most frequent sequence, while ViSpA reconstructs 7 sequences in at least 15% times, and the most frequent sequence is reconstructed 40% times. This plot shows that the found sequences are pretty much reproducible for ViSpA. The third

[4]Experimental results for ShoRAH and ViSpA on HCV data are available in appendix E.

most frequent sequence assembled by ShoRAH is the only sequence among most frequent ShoRAH's assemblies that represents a viable protein sequence.

The plot on Figure 2.20 shows the averaged frequency among all unique matches of 10 most frequent sequences found in bootstrapping tests on 10%-reduced Data1.

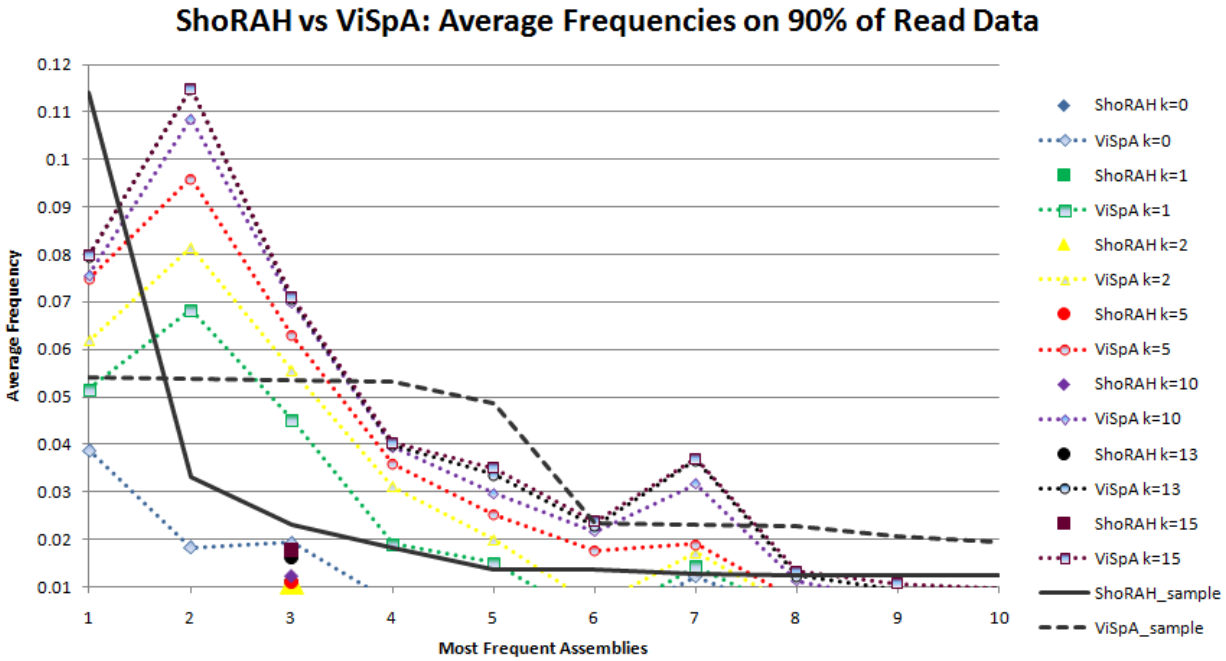

Figure 2.20 *Average frequency among frequencies of matches found in bootstrapping iterations for each of 10 most frequent sequences (unique match). i-th point at x-axis corresponds to the i-th most frequent sequence assembled on the 100% of reads.*

The clustering (or "averaging") during assembling candidate sequences can fix random sporadic genotyping errors - the larger value of *t* erases errors but for the expense of accuracy of the candidate sequences. Unfortunately, the systematic errors will be prone to such method as well as to the method suggested in [22]. A well known drawback of 454 pyrosequencing system - the erroneous indels in homopolymers - may be not an obstacle for assembling a single genome since majority of reads are correct. But quasispecies sequences partition the reads between themselves and some of them may have incorrect reads as a majority. Indeed, top 9 quasispecies reconstructed by ShoRAH contain multiple stop codons in their corresponding amino-acid

sequences, indicating unfixed systematic erroneous indels introduced by 454 Life Sciences machines.

We propose the following way to address homopolymer errors if they happen in the coding region.

1. Find the frame with the start-codon, and align the amino-acid decoded sequence with the one for the reference sequence.
2. In the region immediately preceding stop-codon, find the position *X* where the protein alignment becomes very poor.
3. In the nucleotide quasispecies sequence the position *X* should correspond to a homopolymer which should be either extended or reduced by 1 (the correct fix will make the protein alignment almost perfect).

Usually such errors happen just before "forks", i.e., the place where the quasispecies sequences start significantly deviate from each other. We were able to fix all defective sequences returned by ViSpA using this method.

2.7.3 Analysis of HIV data.

We also compare ViSpA and ShoRAH on HIV dataset, used in the first experiment in [58]. As was said above, we do not preprocess reads with respect to its 454 quality score, and it can explain poorer performance of ShoRAH. Indeed, ShoRAH correctly infers only 2 quasispecies sequences with at most 4 mismatches: one assembly has 3 mismatches with real quasispecies sequence, and the other has 4 mismatches.

ViSpA correctly reconstructs 5 quasispecies with at most 2 mismatches (3 of them among 10 most frequent assemblies): two sequences are inferred without any mismatches (one is among 10 most frequent assemblies), one assembly has 1 mismatch with real quasispecies sequence

(and it is among 10 most frequent assemblies), and the rest sequences have 2 mismatches (one is among 10 most frequent assemblies). The assemblies correspond to a viable protein sequences.

If ViSpA is applied to ShoRAH-corrected reads, it can successfully infer three real quasispecies without any mismatches.

3 GENOTYPE TAGGING WITH LIMITED OVERFITTING

The genetic differences among people can be explored by looking at DNA variations, primarily single nucleotide polymorphisms (***SNP***s). A SNP is a specific DNA locus at which several alleles are observed across at least 1% of population. In the human genome, most of the SNPs are bi-allelic. The major allele is considered to be the wild type whereas the minor allele is called a mutation.

Although SNPs are only around 1% of the 3-billion-base human genome, they can influence human response to a disease, a therapy or different environmental factors, making SNPs valuable for the genome-wide association studies. In these studies, DNA's of infected hosts (cases) and healthy individuals (controls) are searched for SNPs that have significant prevalence among diseased individuals than among controls, determining the genetic risk factor.

Recent advances in next generation sequencing significantly decrease the genotyping cost of genome-wide studies whereas considerably increase the computational complexity of the analysis. Since alleles of neighboring SNPs are often correlated, the genome-associated study can be done on a small subset of SNPs (***informative (tag) SNPs***) that covers the genetic variation of the rest of the data. Tagging saves budget since only tag SNPs are genotyped and, more importantly, reduces complexity of the analysis since the data of the smaller size is used.

Statistical covering of a genetic variation of one SNP by another is usually defined in terms of the correlation between SNPs. It is widely accepted that a variation of a SNP is ***(statistically) covered*** by a variation of another SNP if their correlation r^2 is greater than 0.8.

The problem of tagging is selecting a minimum number of tag SNPs covering all SNP in the given data. Formally, the problem can be formulated as follows.

Minimum Tag Selection Problem (MTS Problem): *given a sample of a population S of n genotypes, each consisting of m SNPs, find a minimum size subset of tag SNPs that statistically covers all other SNPs.*

In practice, a small percentage of SNPs values are missing, reducing the power of an association. A missing SNP value may occur either if this SNP was not initially genotyped or if the SNP value did not pass quality control threshold after typing. When a tag SNP missed a value, the genetic variation of untagged SNPs can be also lost, possibly collapsing distinct genotypes into one. Still correct genotypes can be recovered if several alternative ways are used to cover genetic variation of untagged SNPs [33].

Several approaches were proposed to optimize a number of chosen tags. Based on classification criteria, they can be described as lossy or lossless methods, and as block-based or block-free methods.

Lossless methods yield larger set of tag SNPs since 100% of genetic variation should be captured whereas lossy methods explore trade-off between smaller amount of tags and possible loss of genetic information.

The block-based tagging (e.g., HapBlock [60]) requires a priori information to define "haplotype blocks". A haplotype block is not well-defined, but usually neighboring SNPs are considered to be in one haplotype block if they are in strong linkage disequilibrium [24]. The block-based methods could cover an untagged SNP only by a tag from the same haplotype block, i.e., only within local contiguous sequence of SNPs where the diversity of haplotypes is low. The block-free methods pick tags across the entire chromosome. Such methods include Avi-Itzhak's method of entropy maximization [5], Halldorson's graphic-based algorithm [27], MLR [29], IdSelect [11], STAMPA [28], Tagger [15], BNTagger [37].

To represent genetic variation of an untagged SNP, some tagging methods use a single tag, or multiple tags. The Minimum Tag Selection problem for single-tag methods (e.g., IdSelect [11], Tagger [15]) is equivalent to the Minimum Set Cover problem and can be efficiently approximated within $O(\log m)$ [51]. We showed that for the multiple-tag methods (e.g., MLR [29] and STAMPA [28]), the Minimum Tag Selection problem seems as hard to approximate as the Red-Blue Set Cover problem. Although, IdSelect [11] and Tagger [15] find close to the optimum number of tag SNPs, MLR [29] uses significantly less tags than Tagger since linear combination of tag SNPs can cover a genetic variation of an untagged SNP. However, multiple linear regression might suffer from overfitting since multiple unrelated tags can participate in covering of an untagged SNP. Additionally, the more tags is used by MLR to cover variation of an untagged SNP, the lower prediction accuracy is obtained (see Figure 3.1) [29]. These observations motivate us to explore the trade-off between the number of tags used per non-tagged SNP and possible overfitting in the case when at most two tags can be used.

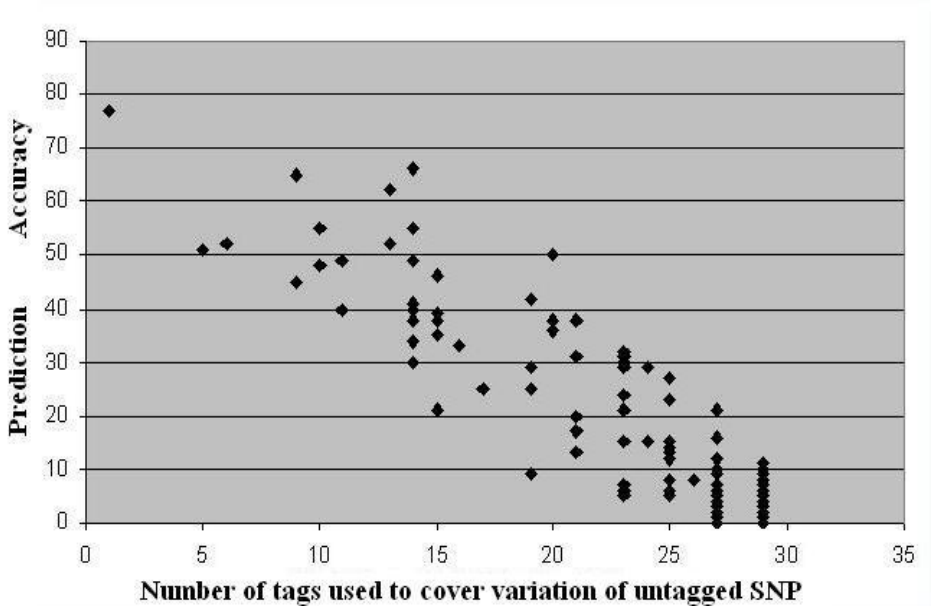

Figure 3.1 *Correlation between the number of tags being used by MLR to cover variation of an untagged SNP and prediction accuracy [29].*

3.1 Correlation for a Triple of SNPs

The section first explains how to measure the correlation between pair of SNPs. Then introduce correlation coefficient for a triple of SNPs. Finally, we discuss how to adjust correlation coefficient to handle missing values.

The correlation coefficient r^2 between two bi-allelic SNPs *A* and *B* measures impact of genetic drift on linkage disequilibrium [30]. Let *D(A, B)* denote linkage disequilibrium betweens SNPs *A* and *B*, and p and q be the frequencies of the major alleles in *A* and *B*, respectively. Then correlation between SNPs *A*, *B* is given as follows:

$$r^2(A,B) = \frac{D(A,B)^2}{p(1-p)q(1-q)}. \quad (3.1)$$

The correlation r^2 reaches 1 only if the allele frequencies of *A* and *B* are the same, and there is an association in haplotypes between the most common alleles at the two SNPs [30].

It would be advantageous to use more than one SNP to cover variation of other SNPs. To the best of our knowledge, nobody proposed how to calculate correlation between one SNP and combination of other SNPs. So here, we introduce the approach to measure correlation between a bi-allelic SNP *A* and a pair of two other bi-allelic SNPs *B* and *C*.

Let $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$ and $\{c_i\}_{i=1}^n$ be the values in SNPs *A*, *B* and *C* on *n* genotypes, respectively. And let 1 code a homozygote major allele, -1 code a homozygote minor allele, and 0 code a heterozygote in each SNP. The best approximation $X = \alpha B + \beta C$ of the SNP *A* by SNPs *B* and *C* is obtained by multiple linear regression method. Geometrically, the best approximation *X* of the SNP *A* by SNPs *B* and *C* can be viewed as the projection of vector $\{a_i\}_{i=1}^n$ on the span determined by the vectors $\{b_i\}_{i=1}^n$ and $\{c_i\}_{i=1}^n$ (see Figure 3.2)).

Obviously, the obtained values of *X* are real numbers, so the rounded best approximation $\bar{X}$ is used. Let $\{\bar{x_i}\}_{i=1}^n$ be the values of a rounded best approximation $\bar{X}$ then

$$\overline{x}_{\iota,i=\overline{1,n}} = \begin{cases} 1, \text{if } (\alpha b_{\mathrm{i}} + \beta c_{\mathrm{i}}) \geq 0.5; \\ -1, \text{if } (\alpha b_{\mathrm{i}} + \beta c_{\mathrm{i}}) \leq -0.5; \\ 0, \text{otherwise.} \end{cases}$$

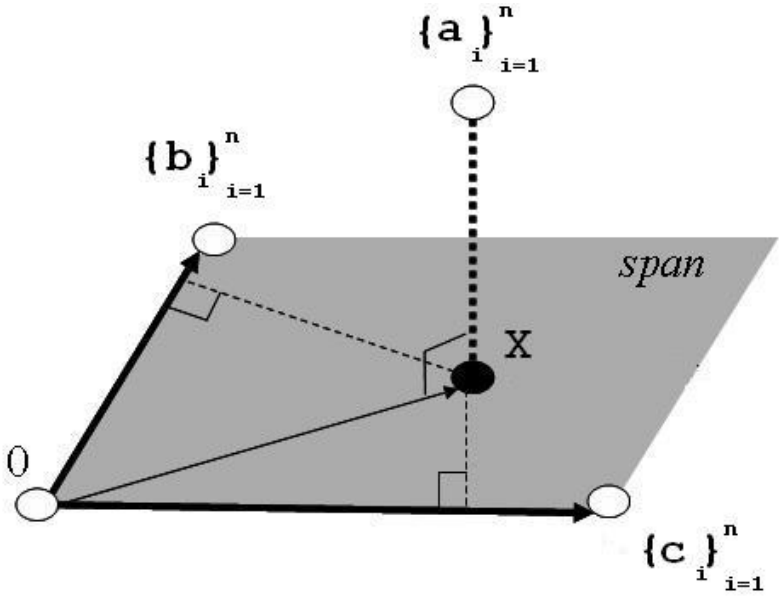

Figure 3.2 *Geometrically, X =αB +βC is the projection of* $\{a_i\}_{i=1}^n$ *on the span determined by the vectors* $\{b_i\}_{i=1}^n$ *and* $\{c_i\}_{i=1}^n$.

A correlation r²(A \|B, C) between SNP A and a pair of SNPs B and C is the correlation between SNP *A* and its rounded best approximation $X = \alpha B + \beta C$ by SNPs *B* and *C*:

$$r^2(A|\, B, C) = r^2(A, X). \tag{3.2}$$

If the correlation $r^2(A|\, B, C)$ for a triple of SNPs *A*, *B* and *C* is greater 0.8 then the genetic variation of the SNP *A* is covered by the SNPs *B* and *C*. In general, the correlation $r^2(A|\, B, C)$ is not symmetric relation since correlation between SNP *A* and SNPs *B* and *C* does not imply, for example, correlation between SNP *B* and SNPs *A* and *C*, or SNP *C* and SNPs *A* and *B*. Thus, the direction of the correlation $r^2(A|\, B, C)$ becomes important.

Further, we assume that missing values are distributed randomly and uniformly. Let $\{a_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$ be values of SNPs *A* and *B* on *n* genotypes, respectively, and let I_{known} in [1, ..., *n*] be the set of indices of genotypes where values in both SNPs *A* and *B* are present. Then we calculate the correlation coefficient as follows:

$$r^2(A,B) = \frac{D(\{a_i\}_{i\in I_{known}}, \{b\}_{i\in I_{known}})^2}{p(1-p)q(1-q)},$$

where p and q are the frequencies of the major alleles of SNPs A and B on the known data for SNP A and known data for SNP B, respectively. This definition does not introduce additional noise since it is based only on available information.

3.2 The Minimum Tag Selection Problem

First, we formulate the Minimum Tag Selection problem when a single tag SNP covers a genetic variation of an untagged SNP. Then the problem is generalized to the case when two tag SNPs are allowed to cover variation of an untagged SNP. Finally, we discuss the complexity of the both problems.

To represent genetic variation of an untagged SNP, a single-tag tagging methods solve the following problem.

Minimum Tag Selection problem for single-tag covering (M1TS Problem): *given a sample of a population S of n genotypes, each consisting of m SNPs, select a minimum size subset of tag SNPs such that genetic variation of each untagged SNP is statistically covered by genetic variation of one tag.*

From graph-theoretic point of view, the M1TS problem can be reformulated as the following problem.

Minimum Dominating Set Problem (MDS Problem): *given a graph* $= (V, E)$, $E \subseteq V^2$, *where each vertex* $i \in V$ *corresponds to SNP i,* $1 \leq i \leq m$, *and edges (i, j) connect highly correlated SNPs i and j (usually,* $r^2(i, j) \geq 0.8$*); find a set* $D \subset V$ *of a minimum cardinality such that every vertex* $i \in V$ *either belongs to D, or is adjacent to vertex* $j \in D$.

The Minimum Dominating Set problem is known to be NP-hard, and unless *P = NP*, it cannot be approximated better than $O(logm)$ via reduction from Set Cover problem [51].

When two tag SNPs are allowed to cover a variation of an untagged SNP, the M1TSP problem is generalized as follows.

Minimum Tag Selection problem for two-tag covering (M2TS Problem): *given a sample of a population S of n genotypes, each consisting of m SNPs, select a minimum size subset of tag SNPs such that genetic variation of each untagged SNP is statistically covered by genetic variation of at most two tags.*

From graph-theoretic point of view, the M2TS problem corresponds to Minimum Dominating Set problem on the following hypergraph $H = (V, E), E \subseteq V^2 \cup V^3$.

Each vertex $i \in V$ corresponds to SNP *i, 1* $\leq i \leq m$. There are two types of hyperedges in hypergraph *H*: hyperedges connecting only two vertices (further referred as edges) and hyperedges connecting 3 vertices. As earlier, a bidirectional edge *(i, j)* connects highly correlated SNPs *i* and *j* (usually, $r^2(i, j) \geq 0.8$). A hyperedge *(i, j, k)* exists if SNP *i* is highly correlated with SNPs *j* and *k*, or, in the other words, their triple correlation is higher than 80%: $r^2(i| j, k) \geq 0.8$. To reduce complexity, if at least one of the edges *(i, j)* or *(i, k)* exists, we do not add hyperedge *(i, j, k)* into the hypergraph *H*. Indeed, high value of the correlation between SNP *i* and SNPs *j* and *k* is due the fact that *i* is highly correlated with one of the other SNPs, *j* or *k*; that is, the hyperedge *(i, j, k)* does not refer to new information. Obviously, since correlation for a triple of SNPs $r^2(i| j, k)$ is not symmetric, the existence of the hyperedges *(j, i, k)* or *(k, i, j)* does not follow from the hyperedge *(i, j, k)* although it implies hyperedge *(i, k, j)*.

Minimum Dominating Set problem on a hypergraph H = (V, E) *asks for a subset* $D \subset V$ *of a minimum cardinality such that every vertex* $i \in V$ *either belongs to D, or is connected by an edge to vertex in D, or is connected by a hyperedge to two vertices in D.*

The Minimum Dominating Set problem on a Hypergraph is NP-hard, however, an integer linear program gives an exact solution for instances of a smaller size. For instances of large sizes, the relaxed linear program returns the approximation of the exact solution (for further details, see Section 3.3).

The M2TS problem cannot be efficiently approximated: it seems as hard to approximate as the following problem.

Red-Blue Set Cover: *given two disjoint finite sets, set R of red elements and set B of blue elements, let* $S \subseteq 2^{R \cup B}$ *be a family of subsets of* $R \cup B$*; find a subfamily* $C \subseteq S$ *that covers all blue elements, but covers the minimum possible number of red elements.*

Indeed, if red elements represent possible tag SNPs and blue elements represent untagged SNPs, the M2TS problem is equivalent to Red-Blue Set Cover. If the set of tag candidates and set of SNPs to cover are not intersected than this special case of M2TS problem is the special case of Red-Blue Set Cover where every set contains only one blue and two red elements. It was shown that unless *P = NP* this special case of the Red-Blue Set Cover cannot be approximated to within $O\left(2^{log^{1-\delta}n}\right)$, where $\delta = \frac{1}{log log^{c}n}$, $\forall c < \frac{1}{2}$ ([16], [12]).

3.3 LP Formulation for M2TS Problem

As follows from the previous section, each instance of M2TS problem can be viewed as an instance of Minimum Dominating Set problem on the corresponding hypergraph. In the section, Boolean Linear Program (BLP) is given for the small instances of the Minimum

Dominating Set problem on the hypergraphs. BLP returns exact solution, but becomes slower for large instances. For a larger size problem, we use BLP relaxation that gives approximate solution.

3.3.1 Boolean Linear Program (BLP)

For a given instance of the Minimum Dominating Set problem on a hypergraph *H = (V, E)* (equivalently, instance of M2TS problem), we formulate Boolean Linear program (BLP) as follows.

We associate boolean variable x_i with each vertex *i* in the hypergraph *H = (V, E)* (or, equivalently, with each SNP *i*). BLP sets x_i to 1 if the vertex *i* is in *D* (e.g., the corresponding SNP *i* is chosen as a tag). Otherwise, x_i is set to 0. Let boolean variables $w_{jk} = x_j \cdot x_k$ be indicators for hyperedges *(i, j, k)* in *E*. The variable w_{jk} indicates whether the hyperedge *(i, j, k)* is chosen by BLP, e.g., vertex *i* is covered by the hyperedge *(i, j, k)*. Indeed, the variable $w_{jk} = 1$ only if both SNPs *j* and *k* are chosen by BLP as tags; otherwise, it is 0.

$$\textbf{Minimize: } \sum_{i=1}^{m} x_i$$

Subject to:

$$x_i + \sum_{(i,j)\in E} x_j + \sum_{(i,j,k)\in E} w_{jk} \geq 1,\ i = \overline{1,m}$$

$$\frac{x_j + x_k}{2} \leq w_{jk} \leq x_j + x_k - 1,\ i = \overline{1,m}$$

$$w_{jk}, x_i \in \{0,1\}, i,j,k = \overline{1,m}$$

The BLP's objective is to minimize the number of vertices *i* in *D* subject to the following constraints. First, we require every vertex *i* either be in *D*, or be connected by edge to a vertex in

D, or be connected by hyperedge to two vertices in D. In the other words, we require SNP *i* be either a tag SNP, or be covered by either a tag SNP or two tag SNPs.

Secondly, BLP should set the indicator variables w_{jk} to 1 if and only if BLP sets to 1 both variables x_j and x_k. Indeed, if x_j or x_k is 0 then $w_{jk} \leq 0$ and BLP sets w_{jk} to 0. Finally, BLP should limit feasible solutions to the boolean variables.

3.3.2 BLP relaxation

BLP gives an exact solution to small instances of the Minimum Dominating Set problem on a hypergraph *H*. For a larger size instances, the BLP relaxation is used.

We use several heuristics to round fractional solutions of the BLP relaxation, including two greedy heuristics and randomized heuristics.

The first greedy heuristic (see Heuristic 1) chooses k variables with the highest weights assigned by BLP relaxation and set them to 1, meaning the corresponding SNPs are chosen as tag SNPs. The value of the parameter k is the smallest integer among all possible such that chosen tag SNPs can cover genetic variation of every untagged SNP in the sample.

Heuristic 1 Greedy heuristics to round BLP solution.

Input

1. $X = \{x_i\}$ is the set of assignments returned by BLP

- *S* is the set of all SNPs
- *F* is the set final assignment

$F \leftarrow \emptyset$

Traverse *X* in decreasing order of assigned values

Let *x* be the current value and *i* is corresponding SNP

$F \leftarrow F \cup \{ x \leftarrow 1 \}$

remove from *S* all SNPs for which *F* can explain all genetic variation

if *S* is empty **then**

break

end if

end of traversing

return *F*

The alternative greedy approach (see Heuristic 2) is iteratively set to 1's the variable(s) x_i with the highest weight assigned at the current run of BLP relaxation. This (these) x_i is (are) explicitly set to 1 in linear program and BLP relaxation is run to obtain new assignments for the rest of variables. The iterative process terminates when all new assignments become zero. Then SNPs that corresponds to all non-zero x_i's, are chosen as tag SNPs. The second heuristic requires slightly more running time than the first heuristic due to several runs of BLP relaxation.

Heuristic 2 Greedy heuristics to round BLP solution.

Input

- *X* is the BLP assignment

Run BLP and obtain assignment *X*

while TRUE **do**

if previously non-fixed variables all assigned to 0 **then**

break

end if

Sort *X* in descending order of values

Fix in *X* to 1 all variables with the highest weight

Run BLP to obtain new assignment for previously non-fixed variables.

end while

return *X*

The third heuristic is based on randomized approach where each SNP is chosen as a tag SNP with the probability, equaled to the weight given by the BLP relaxation.

The BLP relaxation returns an approximation of the exact solution; however, it runs much faster than the BLP and can be used for larger instances of M2TS problem.

3.4 Experimental Results

The section describes several data sets on which we run three tagging methods: 2LR method (BLP and BLP relaxation), MLR and Tagger. We report how well these tagging methods compact the data, measured by a number of tag SNPs. Then we describe how to estimate overfitting in the cross-validation tests and report overfitting for these three methods. Finally, we report data compression of each of three tagging methods if a small percentage of SNP values is missing. Direct comparison with STAMPA is complicated since STAMPA predicts untagged SNPs rather than cover their variation.

3.4.1 Data Sets

We compared 2LR tagging with Tagger [15] and MLR [29] on several publicly available data sets from HapMap. The largest data set has 156 SNPs with minor frequency allele over 0.01%. Each data set has 90 individuals.

Small percentage of missing SNP values was obtained by randomly removing known values from the given data sets.

3.4.2 Data Compression

In this paper, we explore trade-off between the number of tags used to cover a variation of an untagged SNP and possible overfitting of a tagging method. First, we compare how many

tag SNPs each of three tagging methods chooses in the given data sets. The experimental results on HapMap data sets show that 2LR Tagging (BLP and BLP relaxation) is between Tagger and MLR in the number of tags (see Figure 3.3(a)). The BLP usually selects almost the same number

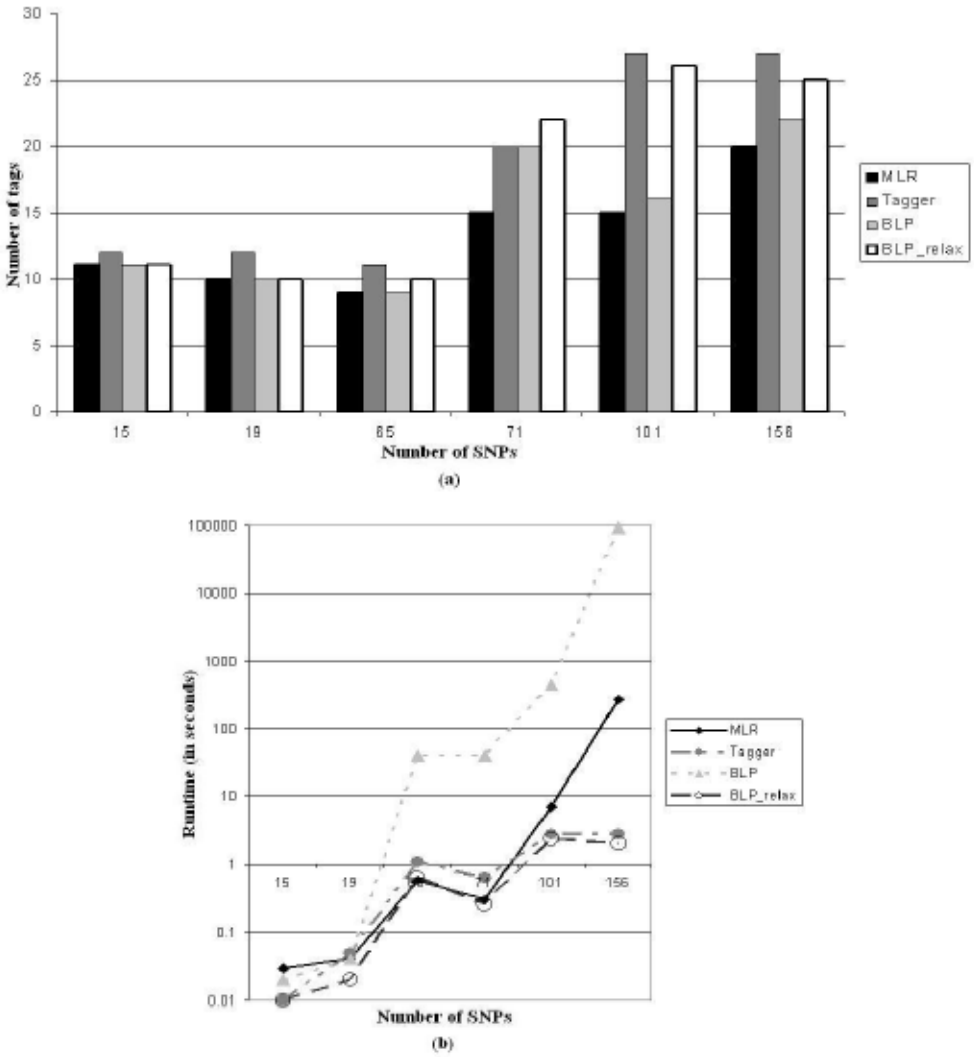

Figure 3.3 *Performance of MLR, Tagger, 2LR tagging (BLP and BLP relaxation) on HapMap data sets. a) For each tagging method, the number of chosen tags is reported for different number of SNPs in the data sets. b) For each tagging method, its runtime is plotted depending on the number of SNPs in data sets. Runtime is given in seconds.*

of tags as MLR. Slight difference in the number of tags between BLP and MLR is due to the limited use of multiple regression in BLP versus unlimited multiple regression in MLR. However, for data sets with more than 100 SNPs BLP becomes significantly slower than the

others methods (see Figure 3.3(b)). The BLP relaxation runs as fast as Tagger and on average finds fewer tags than Tagger but more tags than BLP and MLR due to approximation nature of the relaxation.

3.4.3 Measuring Overfitting via 2-Cross Validation

Further, we compare overfitting of these tagging methods in the cross-validation tests.

In 2-fold cross-validation, we partition individuals in the data into two subsets of (approximately) equal size. We use each subset as a test set and the other subset as the corresponding training set. On each training set, we select tag SNPs and separately compute the average covering correlation on the training set and its test set if the chosen tags are used. If tagging method suffers from overfitting, we will see significant difference between these two averages. To eliminate dependence between an overfitting value and particular partitioning of the sample, we iteratively run 2-fold cross validations on the heavily permuted sample reasonably large number of times. Then the overfitting of tagging method is reported as the averaged difference in covering correlation among all runs.

Experimental 2-fold cross validation results for MLR, Tagger, 2LR tagging (BLP and BLP relaxation) on the data set ADIPOR2-AA with 71 SNPs show that 2-LR Tagging has less overfitting than MLR (on average, 3.6% for 2LR (both BLP and BLP relaxation) versus 5% for MLR), however finds slightly more tags (on average, 19 (21) for BLP (BLP relaxation) and 14 for MLR). On the other hand, 2LR uses less tags than Tagger (19 (or 21 in relaxation) vs. 29 tags on average) while keeping overfitting almost at the same rate (3.2% for Tagger and 3.6% for 2LR (both BLP and BLP relaxation)).

Thus, 2LR tagging finds less tag SNPs than Tagger does. However, 2LR tags have practically the same overfitting as Tagger tags, meaning smaller set of tags can capture the same

amount of a genetic variation as Tagger tags when they are used at the other data sets. The limited used of multiple regression in 2LR tagging rather than unbounded multiple regression in MLR produces slightly larger number of tag SNPs but with smaller overfitting to data. In other words, 2LR tagging finds tag SNPs that can capture more genetic variation than MLR tags with a cost of slight increase in the number of tags.

3.4.4 Missing Data Results

Finally, we explore behavior of three tagging methods on the data set with small percentage of missing data. MLR does not handle missing SNP values and assumes that the data are complete [29]. Thus, we run MLR on the data with imputed missing SNP values [33]. Our experiments show that BLP better tolerates missing data than Tagger (see Table 3.1). The BLP relaxation requires slightly more tags than the BLP to handle missing SNP values. The MLR requires less tag SNPs but the missing values should be imputed first.

Table 3.1 *Comparison of MLR, Tagger and 2LR Tagging (BLP and its relaxation) on ADIPOR2-AA data set with missing SNP values. Number of tags and average correlation is given.*

	No Missing Data	**1% Missing**	**2% Missing**	**3% Missing**
Tagger	20	34	48	49
	0.9549	0.9421	0.925	0.94
MLR	15	15	16	18
	0.984	0.9813	0.989	0.985
BLP	20	24	26	34
	0.945	0.931	0.912	0.8887
BLP relaxation	22	30	45	45
	0.9674	0.9421	0.9235	0.9345

4 CONCLUSIONS AND FUTURE WORK

Next generation sequencing is revolutionizing genomics by inexpensively producing millions of genomic sequences in a single run. Here, we have proposed effective solutions for two inference problems, arisen in particular applications of next generation sequencing, namely, the problem of reconstructing viral quasispecies spectrum from pyrosequencing shotgun reads and problem of inferring informative single nucleotide polymorphisms (SNPs), statistically covering genetic variation of a genome region in genome-wide association studies.

We have proposed and implemented ViSpA, a novel software tool for quasispecies spectrum reconstruction from high-throughput shotgun pyrosequencing reads. ViSpA includes (1) clustering techniques to handle random genotyping errors and rare mutations, (2) haplotype assembler based on maximum-bandwidth paths in weighted read graphs, (3) maximum likelihood estimation of quasispecies frequencies based on EM which takes in account all reads, and (4) separate handling of systematic genotyping errors.

We have validated our method on synthetic and real 454 pyrosequencing shotgun reads from HCV sample and HIV samples. Experimental results show that ViSpA is better in assembling. Although ShoRAH has a superior error correction algorithm, experiments suggest that it is prone to overcorrection, significantly reducing diversity of viral sample.

Additionally, we are exploring extensions of ViSpA to amplicon mode of 454/Roche and to paired-reads. In the latter case, the main difficulty is in selection of pair-aware candidate paths. We also foresee ViSpA application to the analysis of high-throughput sequencing data from bacterial metagenomic samples and ecological samples of eukaryote populations.

ViSpA is available at http://alla.cs.gsu.edu/~software/VISPA/vispa.html in two configurations: one is expected to get values for parameters *n* and *t* whereas the other

configuration automatically set these parameters (for details, see Appendix B). Currently, we also work on extending candidate sets by iteratively applying ViSpA on binary weighted reads. The weights are assigned by EM-based algorithm: it gives 1 to the reads that are not adequately covered by already assembled sequences, and 0 to the reads that are explained by those assemblies. Then ViSpA separately finds superreads for covered and uncovered reads and applies additional fee to the edges that have at least one endpoint among covered reads. Hence, paths, connecting uncovered reads, become more preferable and different candidate set is obtained. Next, EM-based algorithm reassigns new weights to the reads based on the cumulative set of final assemblies. The preliminary results are promising, recovering additional 1-3 sequences after several iterations.

Although ViSpA assembles sequences at the same speed as ShoRAH, we can speed it up via parallelization of some steps, for example, assembling of sequences from the candidate paths.

For the second problem, we propose 2LR heuristic and compare it to Tagger and MLR. Experimental results show that 2LR is between Tagger and MLR in the number of found tags and in overfitting and also better tolerates missing SNP values than Tagger.

In our future research, we will explore whether overfitting of 2LR tagging can be decreased by adding the correlation values between pair or triple of SNPs in BLP and BLP relaxation. Secondly, the correlation between two SNPs does not take into account that 2-SNPs genotypes can be preprocessed by maximum likelihood estimate-based phasing. Finally, since the experiments show that the bounding of tags used for prediction can lower the overfitting, we will search for the optimum number of tags for predicting an untagged SNP in MLR.

REFERENCES

1 454 Life Sciences. http://www.454.com.

2 HIV Databases. http://www.hiv.lanl.gov/.

3 National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov.

4 Quinlan, A.R., Stewart, D.A., Stromberg, M. P., Marth, G.T. (2008). Pyrobayes: an improved base caller for SNP discovery in pyrosequences. *Nature Methods*, 5(2):179-181.

5 Avi-Itzhak, H. I., Su, X., De La Vega, F. M. (2003). Selection of minimum subsets of single nucleotide polymorphisms to capture haplotype block diversity. *Pacific Symposium in Biocomputing*, 2003: 466 - 477.

6 Malde, K., Lanzn, A., Sharma, A., Balzer, S., onassen, I. (2010). Characteristics of 454 pyrosequencing data-enabling realistic simulation with FlowSim. *Bioinformatics*, 26(18): i420 - i425.

7 Bansal, V., Bafna, V. (2008). HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24: i153 - i159.

8 Brinza, D., Zelikovsky, A. (2006). 2SNP: Scalable phasing based on 2-SNP haplotypes. *Bioinformatics*, 22:371 - 373.

9 Alvarez, P., Young, S., Garber, M., Giannoukos, G., Lee, W. L., Russ, C., Lander, E. S., Nusbaum, C., Jaffe, D. B., Brockman, W. (2008). Quality scores and snp detection in sequencing-biosynthesis systems. *Genome Research*, 18(5):763 - 770.

10 MacCallum, I., Kleber, M., Shlyakhter, I. A., Belmonte, M. K., Lander E. S., Nusbaum , C., Jaffee, D. B., Butler, J. (2008). ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Research*, 18:810 - 820.

11 Eberle, M. A., Rieder, M. J., Yi, Q., Kruglyak, L., Carlson, C. S., Nickerson, D. A. (2004). Selecting a maximally informative set of single-nucleotide polymorphisms for association analyses using linkage disequilibrium. *American Journal of Human Genetics*, 74(1):106 – 120.

12 Doddi, S., Konjevod, G., Marathe, M. V., Carr, R. D. (2000). On the red-blue set cover problem. *SODA* 2000:345 - 353.

13 Mark, J., Chaisson, A., Pevzner, P. (2008). Short read fragment assembly of bacterial genomes. *Genome Res*., 18(2):324 - 330.

14 Chen, Z., Fu, B., Schweller, R., Yang, B., Zhao, Z., Zhu, B. (2008). Linear time probabilistic algorithms for the singular haplotype reconstruction problem from SNP fragments, *Journal of Computational Biology*, 15(5):535 - 546.

15 Yelensky, R., Pe'er, I., Gabriel, S.B., Daly, M.J., Altshuler, D., de Bakker, P. I. W. (2005). Efficiency and power in genetic association studies. *Nature Genetics*, 37: 1217 - 1223.

16 Safra, S., Dinur, I. (1999). On the hardness of approximating label cover, *ECCC Report 15*.

17 Lottaz, C., Borodina, T., Himmelbauer, H., Dohm, J. C. (2007). Sharcgs, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17: 1697 - 1706.

18 Martinez-Salas, E., Sobrino, F., de la Torre, J. C., Portela, A., Ortin, J., Lopez-Galindez, C., Perez-Brena, P., Villanueva, N., Najera, R., Domingo, E.(1985). The quasispecies (extremely heterogeneous) nature of viral rna genome populations: biological relevance - a review. *Gene,* 40, pp: 1 - 8.

19 Ruiz-Jarabo, C. M., Martin-Hernandez, A. M., Saiz, J. C., Escarmis, C., Domingo, E., Baranowski, E. (1998). Quasispecies structure and persistence of RNA viruses. *Emerg Infect Dis*, 4:521 - 527.

20 Holland, J. J., Drake, J. W. (1999). Mutation rates among RNS viruses. *In Proceedings of the National Academy of Sciences USA*, 96: 13910 - 13913.

21 Novella, I. S., Weaver, S. C., Domingo, E., Wain-Hobson, S., Clarke D. K., Moya, A., Elena, S. F., de la Torre, J. C., Holland, J. J., Duarte, E. A. (1994). RNA virus quasispecies: significance for viral disease and epidemiology. *Infectious Agents and Disease*, 3: 201 - 214.

22 Eriksson, N., Pachter, L., Mitsuya, Y., Rhee, S. Y., Wang et al. (2008). Viral population estimation using pyrosequencing. *PLoS Comput Biol*, 4:e1000074.

23 H. Fakhrai-Rad, N. Pourmand, and M. Ronaghi.(2002). Pyrosequencing: An accurate detection platform for single nucleotide polymorphisms, Hum Mutat, 19:479 - 485, 2002.

24 Schaner, S. F., Hguyen, H., Moore, J. M., Roy, J., Blumenstiel, B., Higgins J., Gabriel, S. B.(2002). The structure of haplotype blocks in the human genome, *Science*, 296:2225 - 2229.

25 Geraci, F., Genovese, L. M., Pellegrini, M. (2007). A fast and accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. *Algor. Bioinform.*, 4645:49 - 60.

26 von Hahn, T., Yoon, J. C., Alter, H., Rice, C.M., Rehermann, B., Balfe, P., McKeating, J. A.(2007). Hepatitis C virus continuously escapes from neutralizing antibody and t-cell responses during chronic infection in vivo. *Gastroenterology*, 132:667 - 678.

27 Bafna, V., Lippert, R., Schwartz, R., de la Vega, F. M., Clark, A. G., Israil, S., Halldorsson, B.V. (2004). Optimal haplotype block-free selection of tagging SNPs for genome-wide association studies. *Genome Research*, 14:1633 - 1640.

28 Kimmel, G., Shamir, R., Halperin, E. (2005). Tag SNP selection in genotype data for maximizing SNP prediction accuracy. *Bioinformatics*, 21:195 - 203.

29 Zelikovsky, A., He, J. (2007). Informative SNP selection based on SNP prediction. *IEEE Transactions on NanoBioscience*, 6(1): 60 - 67.

30 Kumar, S., Hedrick, P. W. (2001). Mutation and linkage disequilibrium in human mtDNA. *European Journal of Human Genetics*, 9: 962 - 972.

31 de la Torre, J. C., Steinhauer, D. A., Holland, J. J.(1992). RNA virus populations as quasispecies. *Current Topics in Microbiology and Immunology*, 176: 1 - 20.

32 Azimi, N., Skiena, S., Hossain, S. (2009). Crystallizing short-read assemblies around lone Sanger reads. *BMC Bioinformatics* 2009, 10(Suppl 1):S16.

33 Zhang, K., Chen, T., Chao, K.-M., Huang, Y. H. (2004). Approximation algorithms for the selection of robust tag SNPs. *In Procedings of Workshop Algorithms in Bioinformatics*, 3240: 278 - 289.

34 Auch, A. F., Qi, J., Schuster, S. C., Huson, D. H. (2007) Megan analysis of metagenomic data. *Genome Research*, 17: 377 - 386.

35 Reinhardt, J. A., Baltrus, D. A., Hickenbotham, M. T., Magrini, V., Mardis, E. R., Dangl, J. L., Jeck, W. R., Jones, C. D. (2007). Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23: 2942 - 2944.

36 Jojic, V., Laserson, J., Koller. D. (2010). Genovo: De novo assembly for metagenomes, *Research in Computational Molecular Biology*, LNCS 6044/2010: 341 - 356, 2010.

37 Shatkay, H., Lee, P. H. (2006). BNTagger: improved tagging SNP selection using bayesian networks. *Bioinformatics*, 22(14): 211 - 219.

38 Kim, J. H., Li, L., Waterman, M. S.(2004). Haplotype reconstruction from SNP alignment. *J. Comput. Biol.*, 11:507 - 518.

39 Lippert, R., Schwartz, R., Lancia, G., Istrail, S. (2002). Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Brief Bioinform*, 3(1):23 - 31.

40 Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. C. et al. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:376 - 380.

41 Esteban, J. I., Quer, J., Genesca, J., Weiner, A., Esteban, R., Guardia, J., Gomez, J., Martell, M.(1992). Hepatitis C virus (HCV) circulates as a population of different but closely related genomes: quasispecies nature of HCV genome distribution. *Journal of Virology*, 66: 3225 - 3229.

42 Medvedev, P., Brudno, M. (2008). Ab initio whole genome shotgun assembly with mated short reads. *In Proceedings of RECOMB 2008*:50 - 64.

43 Myers, G.(2005). Building fragment assembly string graphs. *European Conf. on Computational Biology*, 79 - 85.

44 Park, J., Noguchi, H., Takagi, T. (2006). Metagene: prokaryotic gene finding from environmental genome shotgun sequences. *Nucleic Acids Research*, 34: 5623 - 5630.

45 Panconesi, A.(2004). Fast Hare: a fast heuristic for single individual SNP haplotype reconstruction. *Lect. Notes Comput. Sci.*, 3240:266 - 277.

46 Prosperi, M., Prosperi, L., Bruselles, A., Abbate, I., Rozera, G., Vincenti, D., Solmone, M. Capobianchi, M., Ulivi, G.(2011). Combinatorial analysis and algorithms for quasispecies reconstruction using next-generation sequencing. *BMC Bioinformatics*, 12(1):5+.

47 Branch, A. D., Fishman, S. L. (2009). The quasispecies nature and biological implications of the Hepatitis C Virus. *Infection*, Genetics and Evolution, 9:1158 - 1167.

48 Holland, J. J., Steinhauer, D. A.(1987). Rapid evolution of RNA viruses. *Annual Review of Microbiology*, 41: 409 - 433.

49 Kurtz, S., Sharma, C. M., Khaitovich, P., Vogel, J., Stadler, P. F., Hoffmann, S., Otto, C., Hackermuller, J.(2009). Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput Biol*, 5(9):e1000502.

50 Sundquist, A., Ronaghi, M., Tang, H., Pevzner, P., Batzoglou, S. (2007). Whole-genome sequencing and assembly with high-throughput, short-read technologies. *PLoS ONE*, 2.

51 Vazirani, V. V. (2001). Approximation Algorithms, Springer-Verlag.

52 Venter, J. C. et al. (2004). Environmental genome shotgun sequencing of the Sargasso sea. *Science*, 304:66 - 74.

53 Sutton, G. G., Jones, S. J. M., Holt, R. A., Warren, R. L. (2007). Assembling millions of short DNA sequences using ssake. *Bioinformatics*, 23: 500 - 501.

54 Westbrooks, K., Astrovskaya, I., Campo, D., Khudyakov, Y., Berman, P., Zelikovsky, A. (2008). HCV quasispecies assembly using network flows. *In Proc. ISBRA*, 2008: 159 - 170.

55 Westbrooks, K. A.(2009). Biological inference using flow networks. *Computer Science Dissertations GSU 2009*, Paper 36, http://digitalarchive.gsu.edu/cs_diss/36

56 Wang, J., Chen, J., Xie, M. (2008). A model of higher accuracy for the individual haplotyping problem based on weighted SNP fragments and genotype with errors. *Bioinformatics*, 24:i105 - i113.

57 Zagordi, O., Geyrhofer, L., Roth, V., Beerenwinkel, N.(2010). Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. *Journal of computational biology*,17(3):417 - 428.

58 Zagordi, O., Klein, R., Dumer, M., Beerenwinkel, N.(2010). Error correction of next-generation sequencing data and reliable estimation of HIV quasispecies. *Nucleic Acids Research*, 38(21):7400 - 7409.

59 Birney, E., Zerbino, D. R. (2008). Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18:821 - 829.

60 Qin, Z., Chen, T., Liu, J. S., Waterman, M. S., Sun, F., Zhang, K. (2005). Hapblock: haplotype block partitioning and tag SNP selection software using a set of dynamic programming algorithms. *Bioinformatics*, 21(1): 131 - 134, 2005.

61 Tammi, M. T., Arner, E., Kindlund, E., Andersson, B.(2003). Correcting errors in shotgun sequences. *Nucleic Acids Research*, 31(15):4663-72.

62 Gajer, P., Schatz, M., Salzberg, S. L.(2004). Automated correction of genome sequence errors. *Nucleic Acids Res.*,32(2):562-9.

63 Chaisson, M. J., Brinza, D., Pevzner, P. A.(2009). De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Res.*, 19(2):336-46.

64 Zhao, X., Palmer, L. E., Bolanos, R., Mircean, C., Fasulo, D., Wittenberg, G. M.(2010). EDAR: An Efficient Error Detection and Removal Algorithm for Next Generation Sequencing Data. *Journal of computational biology*, 17(11): 1-12.

65 Malpani, N., Chen. J. (2002). A note on practical construction of maximum bandwidth paths. *Information Processing Letters*, 83, 3, 175-180.

66 Laird, N. M., Rubin, D. B., Dempster, A. P.(1977). Maximum likelihood from incomplete data via the EM algorithm (with discussions). *Journal of the Royal Statistical Society*, Series B (Methodological) 1977, 39:1–38.

APPENDIX A RELATED PUBLICATIONS / POSTERS

1. Astrovskaya, I., Westbrooks, K., Tork, B., Mangul, S., Mandoiu, I., Balfe, P., Zelikovsky, A, Inferring Viral Spectra from 454 Pyrosequencing Reads. *BMC Bioinformatics*, Proceedings of RECOMB –seq 2011 (accepted).
2. Astrovskaya, I., Zelikovsky, A. (2009). Genotype Tagging with Limited Overfitting. *Proc. of Brazilian Symposium on Bioinformatics*, LNBI 5676:1-12.
3. Westbrooks, K., Astrovskaya, I., Rendon, D. C., Khudyakov, Y., Berman, P., Zelikovsky, A. (2008). HCV Quasispecies Assembly using Network Flows. *Proceedings of Fourth International Symposium on Bioinformatics Research and Applications (ISBRA 2008)*, LNCS 4983: 159-170.

A.1 Oral Presentations

1. Astrovskaya, I., Westbrooks, K., Tork, B., Mangul, S., Mandoiu, I., Balfe, P., Zelikovsky, A. (2011). Inferring Viral Population from Ultra-Deep Sequencing Data, workshop on Computational Advances for Next Generation Sequencing (CANGS 2011) (invited talk).
2. Astrovskaya, I., Westbrooks, and Zelikovsky, A. (2010). Assembling Viral Quasispecies and Estimating Their Frequencies from Ultra-Deep Sequencing Data, Life Tech (invited talk).
3. Astrovskaya, I., Westbrooks, Zelikovsky, A. (2010). Reconstruction of HCV Quasispecies Haplotypes from 454 Life Science Reads, ISBRA 2010 (short abstract).
4. Astrovskaya, I., Zelikovsky, A. (2009). Genotype Tagging with Limited Overfitting. *BSB 2009*.

A.2 Best Poster Awards

1. Astrovskaya, I., Westbrooks, K., Tork, B., Mangul, S., Mandoiu, I., Balfe, P., Zelikovsky, A. (2011). ViSpA: Viral Spectrum Assembling Method, ICCABS'2011.
2. Astrovskaya, I., Zelikovsky, A. (2009). Genotype Tagging with Limited Overfitting, ISBRA 2009.
3. Astrovskaya, I., Babu, D. M., Zelikovsky, A. (2008). 2-LR Tagging: Linear Regression with Controlled Overfitting, MBD Day 2008.

APPENDIX B VISPA DESIGN

B.1 Parameters' Choice

ViSpA automatically sets up the values for the following three parameters:

(1) n =#mismatches allowed between a read and a superread,

(2) m = #mismatches allowed in the overlap between two consecutive reads, and

(3) t = #mismatches expected between a read and a quasispecies.

Note that parameter m depends on the choice of n value. Indeed, since we allow at most n mismatches between read and its superread, it is reasonable to expect up to n mismatches in overlap between consecutive superreads (lower bound) and up to $2n$ mismatches between their subreads. Experiments show that better connectivity in the graph is obtained if m has slightly higher value, equaled to $2.5n$. Since cost on the edge takes into account number of mismatches, edges constructed for $m = n$ are more preferable during path construction then edges, existing only with more than n mismatches in overlap. So for each candidate path selected on the graph with parameter $m = n$, there is either the same path or path with better total cost obtained on the graph with parameter $m = 2.5n$. The latter may happen if path, chosen on the graph with parameter $m = n$, represents part of the quasispecies due to the loss of connectivity in the graph. Thus, parameter m is set to the value $2.5n$[5] in ViSpA.

During the run, ViSpA initially set n to 0 and build a read graph. After selecting candidate paths, ViSpA calculates the portion of paths that represent a quasispecies, rather than its piece due to loss of connectivity. If the portion is no less than 95% of paths, ViSpA considers that value of n is found and proceeds with assembling of candidate sequences out of the chosen candidate paths. Otherwise, n is incremented by 1 and procedure is repeated.

[5] In case of odd n, arithmetic rounding rules are applied.

Since we choose max-bandwidth paths per vertex, ViSpA does not need to transitively reduce a read graph, allowing to remove from implementation the second most time-consuming part of the method (see plot B.1).

As soon as ViSpA finds the value for parameter *n*, it assembles candidate sequences from candidate paths via mutation-based clustering. We vary mutation rate *t/L* from 1.75% up to 8% per sequence (which is in the range observed in [47]) for acute phase of HCV. Since we decrease variability in the data by error correction and clustering reads around superreads, we do not know the diversity in the sample. ViSpA explores parsimonious principle and choose *t* with the smallest number of distinct final sequences.

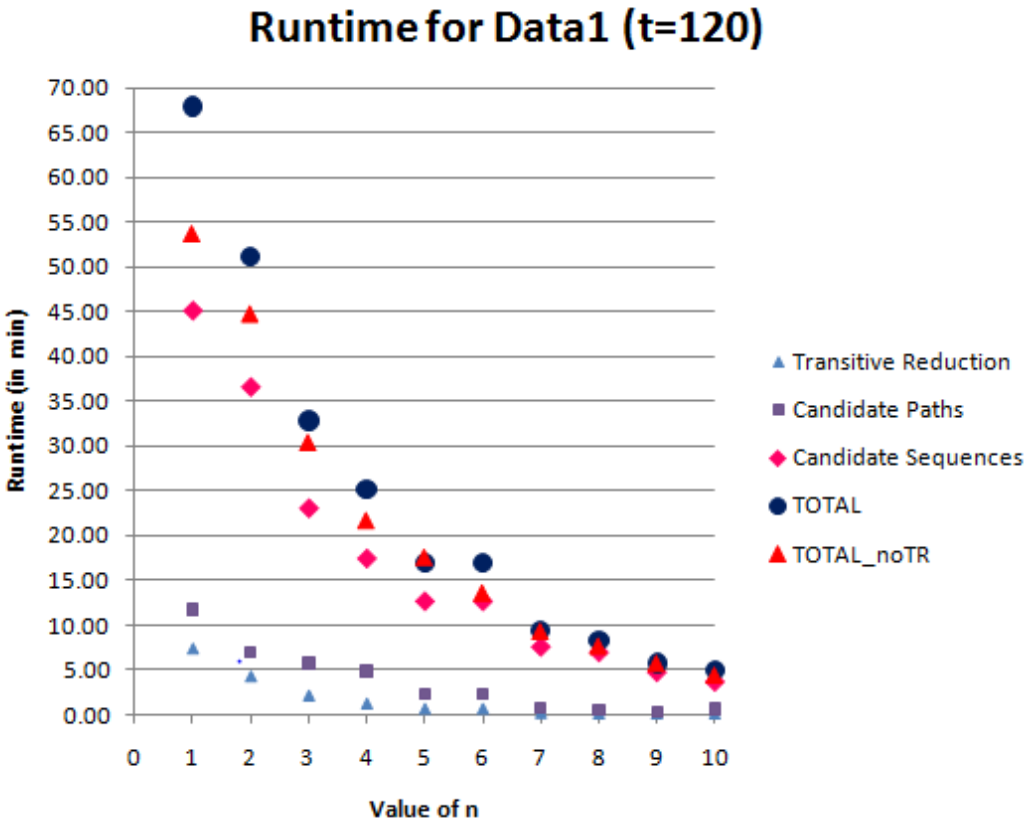

Figure B.1 *ViSpA runtime*[6] *for Data1. All omitted steps (for example, finding superreads) run less than one minute (per each). TOTAL and TOTAL_noTR show the total ViSpA runtime (in minute) with transitive reduction and without it.*

[6] All experiments were run on eight 2.66GHz-CPUs with 8M cache.

B.2 Software Design

ViSpA is a java class library that has the following features:

- Calculate different statistics for aligned read data such as length distribution, position coverage, length distribution of homopolymers, length distribution of insertion and deletions, positions profile
- Correct sequencing errors
- Build read graph
- Select candidate paths and assemble candidate sequences
- Estimate frequency of assemblies by EM on candidate sequences and all reads
- Validate results if viral population is known
- Run bootstrap tests

All classes are in the edu.gsu.cs.qsspcsassmblr package and available at http://alla.cs.gsu.edu/~software/VISPA/vispa.html.

APPENDIX C VISPA: EXPERIMENTAL RESULTS ON SIMULATED ERROR-FREE READS

Table C.1 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

10qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.4103	0.7	0.20214	0.81818	0.99	0.03127	1	1	0.00028	1	1	0.00024
40K	0.5625	0.9	0.1786	0.8788	1	0.0326	1	1	0.00012	1	1	0.000196
60K	0.5014	0.7667	0.2975	0.9091	1	0.025	1	1	0.0013	1	1	0.000067
80K	0.5125	0.8	0.2483	0.9091	1	0.0259	1	1	0.0001596	1	1	0.000091
100K	0.4472	0.7	0.2562	0.8788	1	0.0142	1	1	0.000189	1	1	9.599E-05

Table C.2 *PPV, sensitivity and relative entropy. The results obtained on 20 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

20qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0104	0.0333	0.0267	0.284	0.5167	0.1253	0.6503	0.8333	0.1092	0.8116	0.9333	0.0386
40K	0.0401	0.1167	0.1733	0.3623	0.65	0.1823	0.7067	0.8833	0.0646	0.8656	0.9667	0.0406
60K	0.0554	0.2667	0.2433	0.3949	0.6833	0.1293	0.6933	0.8667	0.1192	0.8	0.9167	0.0611
80K	0.046	0.1333	0.2816	0.334	0.6	0.1813	0.7067	0.8833	0.0548	0.8511	0.95	0.0377
100K	0.0393	0.1167	0.1264	0.433	0.75	0.1555	0.73	0.9167	0.0585	0.8531	0.9667	0.0462

Table C.3 *PPV, sensitivity and relative entropy. The results obtained on 30 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

30qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0154	0.0551	0.0196	0.1205	0.376	0.135	0.485	0.7222	0.0869	0.7411	0.8889	0.0517
40K	0.0243	0.0778	0.1673	0.1859	0.4404	0.2297	0.5116	0.7556	0.0714	0.7629	0.8889	0.0539
60K	0.0216	0.0667	0.0833	0.2593	0.5333	0.1883	0.5233	0.7556	0.0606	0.7852	0.9333	0.0818
80K	0.033	0.1	0.0763	0.2776	0.5556	0.2148	0.5185	0.7778	0.1179	0.7814	0.9111	0.0483
100K	0.0397	0.1111	0.0691	0.2566	0.51111	0.2196	0.5039	0.7444	0.1179	0.736	0.8222	0.0379

Table C.4 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

40qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0101	0.00365	0.0364	0.11	0.3028	0.1755	0.4093	0.6583	0.0975	0.656	0.8417	0.1022
40K	0.01001	0.3583	0.0706	0.1358	0.3487	0.2223	0.4226	0.6883	0.1404	0.6997	0.875	0.0954
60K	0.0125	0.0417	0.0548	0.1502	0.3618	0.2526	0.4366	0.6833	0.1216	0.6887	0.8667	0.0911
80K	0.0181	0.0583	0.0818	0.1832	0.4	0.2133	0.4341	0.675	0.167	0.7055	0.8583	0.0862
100K	0.0126	0.0417	0.013	0.1752	0.375	0.2172	0.4551	0.7167	0.1293	0.7048	0.875	0.0823

Table C.5 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed skewed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

10qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.3818	0.6667	0.3484	0.8788	0.9667	0.0215	1	1	0.000306	1	1	0.000179
40K	0.4704	0.7333	0.2691	0.8788	0.9667	0.0229	1	1	0.0001	1	1	0.00012
60K	0.44	0.67	0.3	0.9091	1	0.0315	1	1	0.00029	1	1	0.000126
80K	0.5556	0.8333	0.2521	0.8788	1	0.0203	1	1	0.000132	1	1	0.000171
100K	0.5778	0.8667	0.2043	0.8788	1	0.004	1	1	0.000099	1	1	0.000071

Table C.6 *PPV, sensitivity and relative entropy. The results obtained on 20 quasispecies followed skewed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

20qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0252	0.097	0.1275	0.3488	0.6333	0.1822	0.6754	0.8667	0.0984	0.8271	0.95	0.0765
40K	0.0557	0.1667	0.19	0.3272	0.6333	0.1169	0.72	0.9	0.0541	0.8235	0.9333	0.0645
60K	0.0454	0.2333	0.1251	0.352	0.6533	0.2104	0.7067	0.8833	0.1547	0.8235	0.9333	0.0481
80K	0.0628	0.1833	0.1112	0.3912	0.6833	0.1609	0.7067	0.8833	0.1446	0.8788	0.9667	0.064
100K	0.0753	0.2167	0.2449	0.4	0.7	0.2172	0.6933	0.8667	0.08	0.8366	0.9333	0.0482

Table C.7 *PPV, sensitivity and relative entropy. The results obtained on 30 quasispecies followed skewed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

30qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0177	0.0948	0.055	0.2118	0.4444	0.2559	0.4673	0.7111	0.1041	0.725	0.8556	0.0437
40K	0.04568	0.1444	0.267	0.2526	0.5222	0.2761	0.4858	0.7444	0.1045	0.7208	0.8889	0.0669
60K	0.0257	0.1068	0.0414	0.2447	0.5	0.377	0.5345	0.7778	0.0716	0.7735	0.9	0.0364
80K	0.0329	0.1	0.1132	0.2653	0.5333	0.2625	0.5231	0.7556	0.0617	0.7757	0.9222	0.0507
100K	0.0434	0.1333	0.2309	0.2895	0.5	0.2475	0.4961	0.7444	0.1853	0.7833	0.9111	0.049

Table C.8 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

40qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0122	0.0575	0.00254	0.1139	0.2856	0.182	0.4175	0.675	0.1582	0.68	0.8667	0.1182
40K	0.0597	0.15	0.1968	0.1432	0.325	0.269	0.3957	0.6333	0.1372	0.7066	0.8833	0.1147
60K	0.02	0.0761	0.0422	0.1624	0.3879	0.2645	0.4296	0.6833	0.143	0.7313	0.8833	0.0664
80K	0.0181	0.0583	0.0626	0.1774	0.436	0.1946	0.4042	0.625	0.1375	0.74	0.9	0.0674
100K	0.0183	0.0583	0.0389	0.1858	0.4	0.2936	0.4539	0.6833	0.1252	0.7455	0.9083	0.0737

Table C.9 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed geometric distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

10qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.4103	0.7	0.20214	0.81818	0.99	0.03127	1	1	0.00028	1	1	0.00024
40K	0.5625	0.9	0.1786	0.8788	1	0.0326	1	1	0.00012	1	1	0.000196
60K	0.5014	0.7667	0.2975	0.9091	1	0.025	1	1	0.0013	1	1	0.000067
80K	0.5125	0.8	0.2483	0.9091	1	0.0259	1	1	0.0001596	1	1	0.000091
100K	0.4472	0.7	0.2562	0.8788	1	0.0142	1	1	0.000189	1	1	9.599E-05

Table C.10 *PPV, sensitivity and relative entropy. The results obtained on 20 quasispecies followed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

20qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0535	0.1667	0.1766	0.1894	0.6333	0.1894	0.6369	0.8167	0.1469	0.776	0.9167	0.1446
40K	0.0429	0.1333	0.1512	0.3724	0.6833	0.2218	0.6841	0.8667	0.1581	0.8261	0.95	0.1002
60K	0.0577	0.1667	0.2173	0.3461	0.6333	0.2173	0.6441	0.8167	0.1628	0.7971	0.9167	0.1299
80K	0.0628	0.1833	0.1127	0.3775	0.6667	0.2446	0.658	0.8333	0.1867	0.8215	0.9167	0.1111
100K	0.0523	0.15	0.1383	0.3621	0.6333	0.287	0.68	0.85	0.102	0.8814	0.9333	0.0715

Table C.11 *PPV, sensitivity and relative entropy. The results obtained on 30 quasispecies followed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

30qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0313	0.1101	0.01312	0.1486	0.4292	0.2288	0.4998	0.7444	0.0611	0.7345	0.8889	0.0838
40K	0.0261	0.0833	0.1468	0.2097	0.4333	0.1973	0.4786	0.7222	0.0639	0.7252	0.8778	0.0867
60K	0.0351	0.1111	0.1215	0.2448	0.4889	0.2711	0.4746	0.7222	0.0679	0.7782	0.9333	0.0778
80K	0.0289	0.0889	0.1003	0.2228	0.4667	0.2966	0.5305	0.7778	0.1396	0.7905	0.9222	0.0789
100K	0.0377	0.1167	0.0931	0.2234	0.4444	0.2291	0.5076	0.7556	0.0983	0.7455	0.8778	0.0377

Table C.12 *PPV, sensitivity and relative entropy. The results obtained on 10 quasispecies followed skewed uniform distribution. The experiments have two parameters: number of produced reads, which varies from 20K up to 100K, and the average read length, which is between 200bp and 500 bp.*

40qsps	avLength=200			avLength=300			avLength=400			avLength=500		
#Reads	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE	PPV	Sensitivity	RE
20K	0.0148	0.05	0.1001	0.1655	0.375	0.1778	0.3959	0.6443	0.1514	0.6825	0.8583	0.1062
40K	0.0153	0.05	0.0467	0.1758	0.4095	0.3004	0.4214	0.6667	0.1327	0.6867	0.8583	0.0932
60K	0.0193	0.0667	0.0604	0.179	0.3833	0.2301	0.4601	0.725	0.1637	0.7094	0.875	0.1074
80K	0.0103	0.0423	0.0502	0.2078	0.45	0.2552	0.4554	0.7083	0.1691	0.7226	0.8667	0.0741
100K	0.0127	0.0417	0.0387	0.211	0.45	0.2478	0.4297	0.6667	0.1669	0.7153	0.875	0.0921

APPENDIX D EXPERIMENTAL RESULTS ON FLOWSIM-GENERATED READS

Table D.1 *Number of times when each of 10 most frequent sequences are repeatedly found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. Reads were generated by FlowSim from 10 HCV quasispecies followed geometric distribution. "N" is the rank of the assemblies. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found with repetition, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set can match the same assemblies obtained on 10%-reduced data.*

	ShoRAH			ViSpA			ShoRAH_reads+ViSpA		
N	k=0	k=1	k=2	k=0	k=1	k=2	k=0	k=1	k=2
1	0.12	0.7	0.86	0.01	0.01	0.01	0.95	0.95	0.95
2	0.45	0.8	0.91	0.01	0.01	0.01	0.95	0.95	0.95
3	0.04	0.59	0.93	0.01	0.01	0.01	0.95	0.95	0.95
4	0.09	0.17	0.59	0.01	0.01	0.01	0.95	0.95	0.95
5	0.02	0.39	0.46	0.01	0.01	0.01	0.95	0.95	0.95
6	0.01	0.17	0.46	0.01	0.01	0.01	0.95	0.95	0.95
7	0.37	0.38	0.38	0.01	0.01	0.01	0.95	0.95	0.95
8	0.04	0.6	0.93	0.01	0.01	0.01	0.95	0.95	0.95
9	0.05	0.17	0.31	0.01	0.01	0.01	0.95	0.95	0.95
10	0.01	0.03	0.12	0.01	0.01	0.01	0.95	0.95	0.95

Table D.2 *Number of times when each of 10 most frequent sequences are repeatedly found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. Reads were generated by FlowSim from 10 HCV quasispecies followed geometric distribution. "N" is the rank of the assemblies. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found without repetitions, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set cannot match the same assemblies obtained on 10%-reduced data.*

	ShoRAH			ViSpA			ShoRAH_reads+ViSpA		
N	k=0	k=1	k=2	k=0	k=1	k=2	k=0	k=1	k=2
1	0.12	0.7	0.86	0.01	0.01	0.01	0.95	0.95	0.95
2	0.45	0.8	0.91	0.01	0.01	0.01	0.95	0.95	0.95
3	0.04	0.59	0.93	0.01	0.01	0.01	0.95	0.95	0.95
4	0.09	0.17	0.59	0.01	0.01	0.01	0.95	0.95	0.95
5	0.02	0.39	0.46	0.01	0.01	0.01	0.95	0.95	0.95
6	0.01	0.11	0.35	0.01	0.01	0.01	0.95	0.95	0.95
7	0.34	0.35	0.35	0.01	0.01	0.01	0.95	0.95	0.95
8	0.01	0.01	0.05	0.01	0.01	0.01	0.95	0.95	0.95
9	0.02	0.05	0.06	0.01	0.01	0.01	0.95	0.95	0.95
10	0.01	0.01	0.01	0.01	0.01	0.01	0.95	0.95	0.95

APPENDIX E EXPERIMENTAL RESULTS ON 454 READS FROM HCV DATASET

Table E.1 *Number of times when each of 10 most frequent sequences are repeatedly found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. "N" is the rank of the assemblies. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found with repetition, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set can match the same assemblies obtained on 10%-reduced data. Note when we run ViSpA on ShoRAH-corrected data, we can repeat each assemblies only 1 time out of 100 iterations. It is evidence that ShoRAH is prone to overcorrection.*

	ShoRAH							ViSpA						
N	k=0	k=1	k=2	k=5	k=10	k=13	k=15	k=0	k=1	k=2	k=5	k=10	k=13	k=15
1	0.01	0.01	0.01	0.03	0.07	0.12	0.13	0.4	0.6	0.76	0.88	0.91	0.95	0.96
2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.22	0.59	0.76	0.9	0.95	0.97	0.97
3	0.41	0.65	0.71	0.71	0.8	0.89	0.92	0.48	0.67	0.78	0.93	0.95	0.97	0.99
4	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.3	0.64	0.78	0.91	0.95	0.97	0.98
5	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.24	0.39	0.49	0.63	0.76	0.85	0.88
6	0.01	0.01	0.01	0.04	0.5	0.9	0.94	0.13	0.17	0.6	0.84	0.94	0.95	0.97
7	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.26	0.3	0.36	0.4	0.64	0.73	0.75
8	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.24	0.3	0.4	0.55	0.75	0.82	0.83
9	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.13	0.24	0.28	0.43	0.83	0.92	0.96
10	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.21	0.33	0.36	0.48	0.66	0.86	0.88

Table E.2 *Average frequency for each out of 10 most frequent assemblies obtained on whole dataset among frequencies of its matches found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. "N" is the rank of the assemblies."OrigFreq" is frequency of the assemblies on the whole data. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found with repetition, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set can match the same assemblies obtained on 10%-reduced data. Note that table contains zero's average frequency. They are due to the precision: if the match was found only in one iteration and frequency was low, then after dividing by 100, it becomes 10^{-5} or less, and due to precision is shown as zero.*

	ShoRAH								ViSpA							
N	OrigFreq	k=0	k=1	k=2	k=5	k=10	k=13	k=15	OrigFreq	k=0	k=1	k=2	k=5	k=10	k=13	k=15
1	0.1142	0	0	0	0.0011	0.0022	0.0041	0.0043	0.0542	0.0388	0.0516	0.0618	0.0749	0.0757	0.0796	0.0797
2	0.0331	0	0	0	0	0	0	0	0.0537	0.0183	0.0684	0.0867	0.1021	0.112	0.118	0.118
3	0.0231	0.008	0.0122	0.0132	0.0132	0.0165	0.0205	0.021	0.0534	0.0611	0.0803	0.0885	0.1089	0.112	0.118	0.1221
4	0.0183	0	0	0	0	0	0	0	0.0533	0.0202	0.0714	0.095	0.1115	0.1206	0.1266	0.1284
5	0.0137	0	0	0	0	0	0	0.0002	0.0486	0.0092	0.0151	0.0199	0.0253	0.0297	0.0335	0.035
6	0.0136	0	0	0	0.0009	0.019	0.0296	0.0304	0.0235	0.0025	0.0033	0.0588	0.0912	0.1077	0.1105	0.1165
7	0.0128	0	0	0	0	0	0	0	0.023	0.0122	0.0143	0.0172	0.0192	0.0321	0.0368	0.0372
8	0.0125	0	0	0	0	0	0	0	0.0229	0.0058	0.0072	0.0092	0.0127	0.0196	0.0226	0.0231
9	0.0124	0	0	0	0	0	0	0	0.0206	0.0025	0.0073	0.008	0.0108	0.0271	0.0398	0.0433
10	0.0124	0	0	0	0	0	0	0	0.0193	0.0049	0.0073	0.0079	0.0105	0.015	0.0364	0.0419

Table E.3 *Number of times when each of 10 most frequent sequences are repeatedly found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. "N" is the rank of the assemblies. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found without repetitions, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set cannot match the same assemblies obtained on 10%-reduced data. Note when we run ViSpA on ShoRAH-corrected data, we can repeat each assemblies only 1 time out of 100 iterations. It shows that ShoRAH is prone to overcorrection.*

	ShoRAH							ViSpA						
N	k=0	k=1	k=2	k=5	k=10	k=13	k=15	k=0	k=1	k=2	k=5	k=10	k=13	k=15
1	0.01	0.01	0.01	0.03	0.07	0.12	0.13	0.4	0.6	0.76	0.88	0.91	0.95	0.96
2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.22	0.59	0.75	0.88	0.94	0.97	0.97
3	0.35	0.54	0.6	0.6	0.64	0.74	0.79	0.23	0.55	0.69	0.81	0.89	0.91	0.93
4	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.15	0.37	0.62	0.75	0.86	0.87	0.89
5	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.24	0.39	0.49	0.63	0.76	0.85	0.88
6	0.01	0.01	0.01	0.01	0.03	0.04	0.06	0.09	0.13	0.25	0.59	0.71	0.77	0.81
7	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.26	0.3	0.36	0.4	0.63	0.73	0.75
8	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.14	0.17	0.23	0.31	0.51	0.57	0.62
9	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.06	0.12	0.12	0.19	0.32	0.43	0.5
10	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.15	0.22	0.25	0.36	0.41	0.45	0.45

Table E.4 *Average frequency for each out of 10 most frequent assemblies obtained on whole dataset among frequencies of its matches found among the 10 most frequent quasispecies assembled on the 10%-reduced set of reads. "N" is the rank of the assemblies. "OrigFreq" is frequency of the assemblies on the whole data. If two sequences have less than k mismatches, they are considered indistinguishable, that is, matches to each other. Match is found without repetition, that is, two candidate sequences from ten most frequent assemblies obtained on whole data set cannot match the same assemblies obtained on 10%-reduced data. Note that table contains zero's average frequency. They are due to the precision: if the match was found only in one iteration and frequency was low, then after dividing by 100, it becomes 10^{-5} or less, and due to precision is shown as zero.*

	ShoRAH								ViSpA							
N	OrigFreq	k=0	k=1	k=2	k=5	k=10	k=13	k=15	OrigFreq	k=0	k=1	k=2	k=5	k=10	k=13	k=15
1	0.1142	0	0	0	0.0011	0.0022	0.0041	0.0043	0.0542	0.0388	0.0516	0.0618	0.0749	0.0757	0.0796	0.0797
2	0.0331	0	0	0	0	0	0	0	0.0537	0.0183	0.0684	0.0814	0.0961	0.1085	0.115	0.115
3	0.0231	0.0068	0.01	0.011	0.011	0.0123	0.0163	0.0177	0.0534	0.0193	0.0452	0.0556	0.063	0.0698	0.0706	0.0711
4	0.0183	0	0	0	0	0	0	0	0.0533	0.0067	0.019	0.031	0.0359	0.0395	0.0399	0.0403
5	0.0137	0	0	0	0	0	0	0	0.0486	0.0092	0.0151	0.0199	0.0253	0.0297	0.0336	0.035
6	0.0136	0	0	0	0	0.0004	0.0005	0.0011	0.0235	0.0016	0.0024	0.006	0.0177	0.0216	0.0229	0.0238
7	0.0128	0	0	0	0	0	0	0	0.023	0.0122	0.0143	0.0172	0.019	0.0316	0.0367	0.037
8	0.0125	0	0	0	0	0	0	0	0.0229	0.0036	0.0043	0.0055	0.0071	0.0112	0.0124	0.0132
9	0.0124	0	0	0	0	0	0	0	0.0206	0.0011	0.0023	0.0023	0.0037	0.007	0.0093	0.0106
10	0.0124	0	0	0	0	0	0	0	0.0193	0.0034	0.0049	0.0055	0.0079	0.0088	0.0097	0.0097