ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lebedeva** Jméno: **Anastasia** Osobní číslo: **406468**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Bioinformatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozšíření populačního genotypového fázování o fragmentové fázování

Název diplomové práce anglicky:

Extension of genotype population-based phasing with

Pokyny pro vypracování:

"Cílem práce je rozšířit metody pro population-based genotype phasing zahrnutím informace o přečtených fragmentech (read-based phasing).
Konkrétně je cílem prozkoumat existující modely a algoritmy pro genotype phasing a analyzovat vhodnost dostupných dat z 2nd-gen a 3rd-gen sequencing pro vybrané read-based metody, popsat Li-Stephens model fázování [1] a rozšířit jej o informace ze čtení fragmentů a navrhnout a implementovat algoritmus (samostatně či jako součást jiného programu), který by efektivně prováděl fázování (a případně i genotype refinement) pomocí
tohoto modelu.Výsledky algoritmu na několika typech dat budou srovnány s
jinými existujícími řešeními pro read- a population-based phasing.
U výsledku je kladen důraz na přesnost před rychlostí."

Seznam doporučené literatury:

[1] Li, N. and Stephens, M. "Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data." Genetics
165 (2003): 2213?2233
[2] Loh, Po-Ru, Pier Francesco Palamara, and Alkes L Price. ?Fast and
Accurate Long-Range Phasing in a UK Biobank Cohort.? Nature genetics
48.7 (2016): 811?816
[3] M. Patterson, T. Marschall, N. Pisanti, L. van Iersel, L. Stougie, G. W. Klau,
A. Schönhuth: "WhatsHap: Weighted Haplotype Assembly for
Future-Generation Sequencing Reads" Journal of Computational Biology, 22(6)
(2015): 498 ? 509
[4] Delaneau O, Zagury J-F, Marchini J. Improved whole-chromosome
phasing for disease and population genetic studies. Nature Methods 10.
(2013): 5?6

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Tomáš Gavenčiak, Ph.D.,

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.01.2018** Termín odevzdání diplomové práce: _____

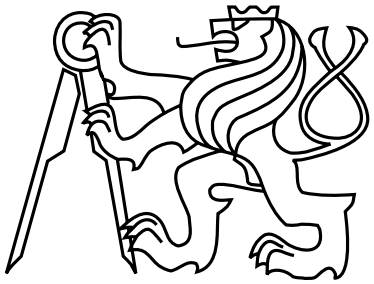
Platnost zadání diplomové práce: **30.09.2019**

[redacted]
Mgr. Tomáš Gavenčiak, Ph.D.
podpis vedoucí(ho) práce

[redacted]
podpis vedoucí(ho) ústavu/katedry

[redacted]
prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Haplotype Estimation using Sequencing Reads

Bc. Anastasia Lebedeva

Supervisor: Mgr. Tomáš Gavenčiak, Ph.D

Study Programme: Open Informatics

Field of Study: Bioinformatics

May 17, 2018

Aknowledgements

I would like to thank my thesis advisor Mgr. Tomáš Gavenčiak, Ph.D of the Faculty of Mathematics and Physics at Charles University. He consistently allowed this thesis to be my own work, but steered me in the right direction whenever he thought I needed it.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 15, 2018 .....

Abstract

The term haplotype refers to a group of alleles in an organism that is inherited together from a single parent. Although the majority of genetic studies requires knowledge of haplotypes, available DNA sequencing technologies do not produce the information explicitly. For that reason, various computational methods for haplotype estimation (also known as phasing) have been actively developed.

Some of the currently available sequencing platforms produce fragments spanning long regions of original DNA. As soon as a fragment spans more that one heterozygous variant, this information can be efficiently utilized during phasing. However, phasing based only on reads does not achieve sufficient accuracy due to the high sequencing error rate longer reads suffer from. At the same time, population-based phasing methods which are highly accurate and constitute current state-of-the-art are often computationally too expensive.

Recently, Eagle2 population-based phasing algorithm significantly outperformed available phasing software in terms of run time requirements without loss in accuracy. In this work we present a new phasing algorithm which adopts main features of the Eagle2 algorithm and extends the method by read-based phasing. This allows to increase phasing accuracy without increasing time complexity. It also allows to perform phasing on a smaller reference panel.

Keywords: Haplotype estimation, Phasing, Read-based phasing

Abstrakt

Haplotypem rozumíme skupinu alel, které jsou přenášeny společně. Přestože většina genetického výzkumu vyžaduje znalost haplotypů, v současnosti dostupné technologie sekvenování DNA tuto informaci explicitně neposkytují. Z tohoto důvodu probíha aktivně vývoj výpočetních metod pro odhad haplotypů (neboli fázování).

Některé z aktuálně používaných platforem pro sekvenování produkují fragmenty pokrývající dlouhé úseky původní DNA. Kdykoli fragment pokrývá více než jednu heterozygotní variantu, lze tuto informaci efektivně využít během fázování. Fázování pracující pouze s fragmenty DNA nicméně nedosahuje požadované přesnosti kvůli vysoké chybovosti těchto platforem. Na druhou stranu, metody populačního genotypového fázování, které dosahují vysoké přesnosti, jsou často výpočetně příliš náročné.

V nedávné době Eagle2 (algoritmus populačního genotypoveho fázování) v rychlosti výrazně překonal dosud dostupné fázovací algoritmy bez ztráty přesnosti. V této práci představujeme nový algoritmus fázování, který kombinuje vlastnosti Eagle2 se schopnostmi fragmentového fázování. Tímto je umožněno zvýšení přesnosti bez nárůstu časové složitosti a zároveň dovoluje operovat na menším referenčním panelu.

Klíčová slova: Odhad haplotypů, Fázování, Fragmentové fázování

Contents

I Introduction **1**

1 Background **3**
1.1 Biological background . . . 3
1.1.1 Genetic variation . . . 3
1.1.2 Haplotype . . . 4
1.1.3 Genetic recombination . . . 4
1.1.4 Genome sequencing . . . 5
1.2 Data formats . . . 6

2 Problem statement **9**
2.1 Phasing problem . . . 9
2.1.1 The importance of phase information . . . 9
2.1.2 Ambiguity and Data sources . . . 10
2.2 Related works . . . 12
2.2.1 Population based phasing . . . 12
2.2.2 Read-based phasing . . . 13
2.2.3 Hybrid phasing . . . 13
2.3 Objectives . . . 14
2.4 Student contribution . . . 15
2.5 Outline . . . 15

II Materials and Methods **17**

3 Eagle2 phasing algorithm **19**
3.1 Algorithm overview . . . 19
3.1.1 Step 1: HapHedge data structure construction . . . 19
3.1.2 Step 2: Exploration of diplotype space . . . 20
3.2 Proposed extension . . . 20

4 Hybrid phasing algorithm **21**
4.1 Preliminaries . . . 21
4.2 Diplotype probability model . . . 21
4.2.1 Haplotype probability model . . . 22
4.2.2 Read model . . . 22
4.3 Haplotype inference . . . 23

4.3.1 Hidden Markov model . . . 23
4.3.2 State space exploration . . . 23

5 Implementation **25**
5.1 Tool for Read Panel Construction . . . 25
5.1.1 The .var format . . . 26
5.1.2 Execution . . . 26
5.2 Phasing Algorithm . . . 28
5.2.1 Step 1: Read panel construction . . . 28
5.2.2 Step 2: PBWT Index construction . . . 28
5.2.3 Step 3: Haplotype inference . . . 28
5.3 Computational cost . . . 29
5.3.1 Discussion . . . 30
5.4 Execution . . . 31

III Results **33**

6 Testing design **35**
6.1 Testing scenarios . . . 35
6.2 Data . . . 35
6.2.1 Data sources . . . 35
6.2.2 Dataset simulation . . . 36
6.3 Performance evaluation . . . 36
6.3.1 Environment . . . 36
6.4 Number of HMM states . . . 36

7 Results and discussion **39**
7.1 Results . . . 39
7.1.1 Scenario 1: Impact of read panel . . . 39
7.1.2 Scenario 2: Number of HMM states . . . 39
7.1.3 Scenario 3: Low coverage . . . 40
7.1.4 Scenario 4: High sequencing error rate . . . 41
7.1.5 Read panel construction . . . 41
7.2 Discussion . . . 42
7.3 Conclusion . . . 44
7.4 Future work . . . 45

List of Figures

1.1 Example of variations which could be classified as SNPs. The SNP on the left side represents a substitution of nucleotide C to A; the SNP on the right side is an example of INDEL, which is an insertion of two nucleotides AA. Note that we illustrate two single *3'* to *5'* strands instead of double strands for simplicity. 4
1.2 Illustration of a recombination event between two homologous chromosomes (one coming from mother, another from father) happening during meiosis. . . 5

2.1 Toy example illustrating phasing problem input and output.
($*$) Note that we do not actually know which hapotype came from the father and which from the mother, but we know that each came from a different parents. That is why alleles of the first ambiguous variant is randomly assigned to one of the haplotypes. 10
2.2 Toy example illustrating how phase information is preserved in reads. 11

5.1 Example of the *.var* format. 26

6.1 Box plots of the relation between the number of HMM states and the switch error rate and time, respectively. 37

7.1 Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 15 and for $P = 100$. 40
7.2 Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 15 and for $P = 100$ and $P = 1000$. 41
7.3 Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 5 and 15 and for $P = 100$. 42
7.4 Box plots of the relation between the read set parameters (length and sequencing error rate) and *.var* creation time for the dataset with depth 15 and 5, respectively. 43
7.5 Histogram of distances between the start and end of switch error coordinates in the genome of sample $HG00099$ made by the phasing algorithm when no read panel was given and when a dataset with reads of length $1k$, $5k$ and $10k$ and 1% sequencing error rate was given. 44

List of Tables

1.1 Main characteristics of available platforms for genomic data sequencing reported in [KGE].
(∗) An interval represents range of average values producing by different instruments of the platform; a single value represents average for the platform. . 6
1.2 A brief description to the most common bioinformatics file formats. 8

Part I

Introduction

Chapter 1

Background

With recent technological advances, enormous amounts of genotype data are being generated. However, the majority of information in these data is best exploited through phased haplotypes, which identify the alleles that were inherited together from a single parent.

The wide range of application of inferred haplotype phase information includes, among other, characterizing the relationship between variants and disease susceptibility [TBT$^+$], imputing low frequency variants [MHM$^+$], [BBd], modeling a population separation history [SSEK] and inferring recombination models [Kea]. This comprehensive list makes haplotype phasing an important preliminary step in majority of genetic studies.

The necessity of phasing is apparent as the currently available technologies do not read the genome as a whole, but break it into many short fragments, which discards desired phase information. The difficulty of the problem also stems from the diversity and complexity of life, described by the laws of genetics and inheritance. This chapter is aimed at introducing the problem essentials from biological, technological and computational points of view.

1.1 Biological background

1.1.1 Genetic variation

There are no two humans, except for identical twins, who are genetically identical. However, on average, each human is 99.9% similar to any other human [Chi]. A common scenario of how the difference occurs is by propagating of a harmless mutation, which may be a silent or even a positive mutation, to the following generations. Harmful mutations are propagated in the same way but are less likely to survive natural selection.

Specific locations of differences between individuals are classified are generally referred to as *variants*. In its simplest form, a variation has several different spellings, referred to as *alleles* [FGC]. Genetic variants are classified based on whether DNA material was substituted, gained (duplicated or inserted), lost or rearranged (inverted, translocated) [HS]. The most important for us will be *single-nucleotide polymorphisms* or *SNPs*. Those include variations in a single nucleotide and small insertions and deletions (also called *indels*) ranging from 1 to 10000 base pairs (bp) [MMPD]. Fig. 1.1 illustrates variants classified as SNPs. Importantly, a variant is classified as SNP when it is present to some appreciable degree within a population (e.g. over 0.1%).

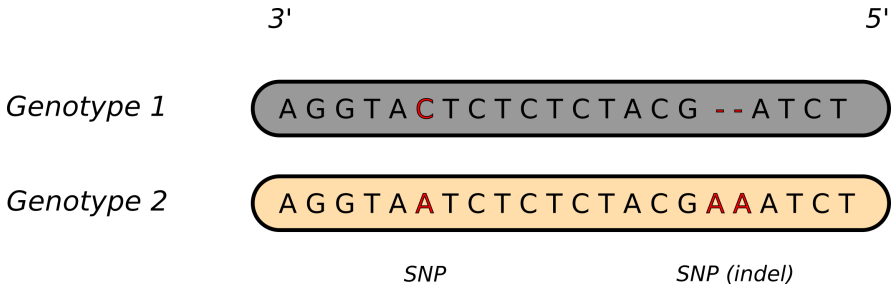


Figure 1.1: Example of variations which could be classified as SNPs. The SNP on the left side represents a substitution of nucleotide C to A; the SNP on the right side is an example of INDEL, which is an insertion of two nucleotides AA. Note that we illustrate two single *3'* to *5'* strands instead of double strands for simplicity.

Presence of variations along with the extensive size of genomic data forced scientists to construct so-called *reference genome*, which is an idealized assembly derived from the DNA of a number of people. In order to address a variation, scientific community compares it to the reference and announces matching with the reference or matching with an alternative of the gene in the locus.

1.1.2 Haplotype

Humans are called diploid organism because they have two alleles at each genetic locus, with one allele inherited from each parent. Each pair of alleles represents the genotype of a specific gene. A genotype is *homozygous* at a particular gene locus if both alleles are identical and is *heterozygous* if the alleles differ.

The set of alleles inherited from a single parent is referred to as *haplotype* [CML16], [Boa12]. Therefore, one might think of *haplotype estimation* (or *phasing*) to be splitting alleles observed in an organism to several groups. The number of the groups is given by the organism *ploidy*, which is defined as a number of complete sets of chromosomes in a cell. In case of humans, the number is two.

1.1.3 Genetic recombination

Each haplotype is a novel and unique combination of alleles, produced before being passed from a parent to its offspring during the process called *genetic recombination*. In eukaryotic cells, this process takes place during meiosis and involves genetic information exchange between homologous chromosomes. One of the possible results of a recombination event is laid out in Fig. 1.2. Four daughter cells are produced during one meiosis (each is haploid and has a unique combination of the parental cells genome) and only one of those will be passed to the offspring. Consequently, genetic recombination is the primary process contributing to genetic variance and describing how genetic information is passed between generations.

Therefore, we expect that related individuals share common segments of DNA. In genetic genealogy, those segments are called to be *identical by descent (IBD)*. Their length depends on the number of generations since the most common ancestor at the locus of segment [Tho],

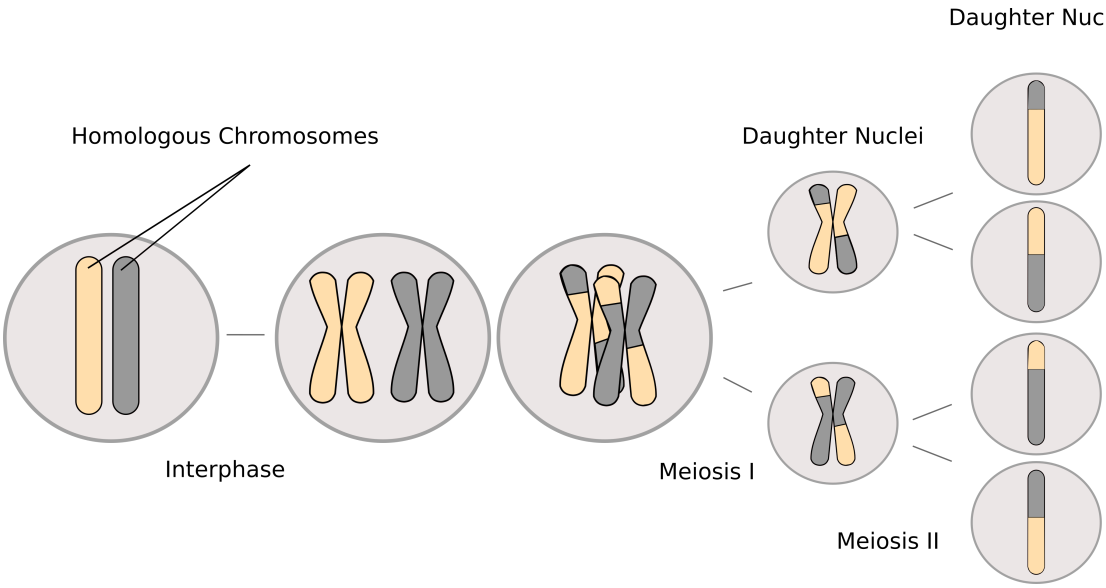


Figure 1.2: Illustration of a recombination event between two homologous chromosomes (one coming from mother, another from father) happening during meiosis.

due to the essence of the recombination event. Many studies conducted on population rely upon presence of the IBD patterns.

1.1.4 Genome sequencing

An organism genome sequencing is performed to access its genetic information. Formally, *DNA sequencing* is a process of determining the precise order of nucleotides within a DNA molecule. Currently available technologies generate a set of *reads* (fragments of original DNA), instead of reading a complete genome as a whole.

Sequencing techniques are commonly divided into generations as follows [HC]:

- *First-generation* methods enabled sequencing of clonal DNA populations.
- *Second-generation* methods massively increased throughput by parallelizing many chemical reactions.
- *Third-generation* methods attempting direct DNA sequencing at the single molecule level.

The first generation mainly refers to the Sanger method, which in mass production form was the technology which produced the first human genome in 2001. However, the cost and time was a major stumbling block. The short span of years starting in 2005 has marked the emergence of a new generation of sequencers to break the limitations of the first generation [KGE]. Nowadays, first-generation sequencing is almost completely substituted with the second and the third-generations sequencing technologies.

Platform	Read length (bp)*	Error rate (%)*	Advantages	Disadvantages
First Generation			*Low error rate*	*Cost, speed*
ABI Sanger	400-900	0.3		
Second Generation			*Cost, speed (massively parallel)*	*Short reads*
454	100-700	1		
Illumina	150-300	0.1-1		
SOLiD	75	0.1		
Ion Torrent	200-400	1		
Third Generation			*Cost, speed, availability, long reads*	*High error rate*
PacBio	1.3k-15k	12-15		
Oxford Nanopore	10k	12		

Table 1.1: Main characteristics of available platforms for genomic data sequencing reported in [KGE].
(∗) An interval represents range of average values producing by different instruments of the platform; a single value represents average for the platform.

The second-generation methods require breaking long strands of DNA into small segments. After the breaking, those fragments are amplified and synthesized [Ill]. The process of synthesis is carefully monitored, for instance with a usage of fluorescent molecules, and the original DNA fragment is thereby read. Massive parallelization of the processes make the technology much faster and less costly in comparison with the first-generation technologies.

Instead of breaking DNA on fragments, the third-generation sequencing works by reading the nucleotide sequences at the single molecule level. This results in substantially longer reads than methods of the previous generations were capable to produce. The difference also influences time requirements of the methods, cost, and amount of sequencing errors.

An overview of the main features of the popular sequencing platforms of the first, second and third generations are presented in the Tab. 1.1.

While being still under active development, the third-generation technologies suffer from high error rate, as it can be observed from Tab. 1.1. However, the methods have a great potential and are expected to bring an essential impact in the near future.

1.2 Data formats

An organism genome sequencing results in one or multiple *.fastq* files, containing so-called raw sequencing data. It includes multiple sequences of the four nucleotides *A, C, T, G* (in case of DNA sequencing) along with estimated sequencing qualities for each base encoded as single byte ASCII codes (see Tab. 1.2 for details).

In order to operate with the information, one is required to align the obtained reads

to a reference sequence, which is commonly stored in *.fasta* format (see Tab. 1.2). The alignment results in a file in *.sam* format, which contains estimated positions of the reads in the original genome and the alignment qualities for each read's base. Commonly, the next step in genetics studies is variant calling, which result in a *.vcf* file. This allows to identify SNPs and other larger polymorphisms, that the sequenced genome possesses.

Among other, a *.vcf* file contains information specifying weather observed genotype is equal to the reference allele (denoted as *zero*) or to the alternative allele (denoted as *one*), or both, for each listed position for each present sample. Genotype containing both *zero* and *one* in a marker would mean that the sample is heterozygous in the variant. It might also happen that more than two alleles would be observed in a population at a specific marker. In this case, *.vcf* would contain a list of the alternatives listed in *ALT* column (see Tab. 1.2 for details). In the case of a multialelic site integers are used to refer to one of the alleles (corresponding to the allele position in the list), analogically to *one*, used to denote an alternative in case of a bialelic site.

There are plenty of other bioinformatics data formats, e.g. *CRAM, ORC, Avro* etc. The mentioned ones are those considered the most common and those which we primary access during the study. In further text, we describe one more data format which we designed in order to optimally utilize haplotype information preserved in reads which span more than one variant. Details of the format are present in Sec. 5.1.1.

Format	Content
.fasta	• header starting with '>', containing a sequence name • *>reference_ chr20* • sequence • *ATAGACATAGA..*
.fastq	**For each read contains:** • name • *@read1* • sequence • *ATAGACATAGA..* • base qualities (encoded as single byte ASCII codes) • $!*((((****+)..$
.sam/.bam	• header lines starting with '@' • format version • sort order of alignments etc. **For each read contains:** • information about the alignment • read' position • mapping quality • FLAG - set of additional information describing the alignment • TAGs - additional optional information • mate pair/paired end reads information • mate position • inner size etc. • quality and alignemnt denoted by mapping/pairing • cigar - reference to read subsequence and associated mapping operation (matched, clipped etc.) • read's information • name • sequence • bases' qualities as Phred Quality Scores
.vcf/.bcf	• meta-information in a form *##key=value*, where • key is e.g. *INFO, FILTER, FORMAT* or other • header naming 8 mandatory columns: *#CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO* **For each called variant contains:** • variant describing fields, which correspond to mandatory header columns • *20, 14370, rs6054257, G, A, 29, PASS, NS=3; DP=14; AF=0.5* • genotype fields for each present samples, which correspond to header optional field *FORMAT*. When *FORMAT* is not specified, the field contains genetic information only • 0|1 - in case of phased variant • 0/1 - in case of unphased variant

Table 1.2: A brief description to the most common bioinformatics file formats.

Chapter 2

Problem statement

2.1 Phasing problem

We define phasing as a process of haplotype estimation from genomic data. Currently available sequencing technologies do not allow to gain haplotype information directly from a biological experiment with desirable efficiency, as it was discussed in the previous section. For the reason various computational methods are applied to estimate haplotypes (e.g. [SSD], [SS], [LDPea], [BBc] etc.).

Generally, the input of a phasing software is a *.vcf* file, with contains information about alleles the phasing sample has at each polymorphic position along the genome. However, mutual information about the markers' states is lost. Therefore, a phasing software aims at determining which alleles appear together on the same chromosome, i.e. estimating haplotype information. Henceforth we consider a human diploid genome and refer to the two haplotypes as *maternal haplotype* and *paternal haplotype*, we denote as h^{α} and h^{β} respectively.

2.1.1 The importance of phase information

The importance of phase information for human genomics might not be obvious. However, many findings — both from recent studies and in the more established medical genetics literature — indicate that relationships between genotype and phenotype, including genetic disease, can be better understood with phase information [TBT^{+}].

For instance, there are clinical conditions and disorders influenced by compound heterozygosity within a single genes, e.g. *Cerebral Palsy*, *Hemachromatosis*, *Mediterranean Fever* and many others [TBT^{+}]. Since most of mutations are inherited from parents, sequencing of the parental genomes might be performed to estimate phase information. However, it is never guaranteed that phase information inferred from parental genomes will be univocal. For instance, in case when both parents are heterozygous in markers of interest, it is not clear which alleles were inherited together. Meanwhile, computationally methods for haplotype estimation allow diagnosing the conditions with no need to sequence parental genomes.

Phase information is also important in the population studies. For instance, it was demonstrated in [NLS] that greater differentiation of human populations can be obtained by ex-

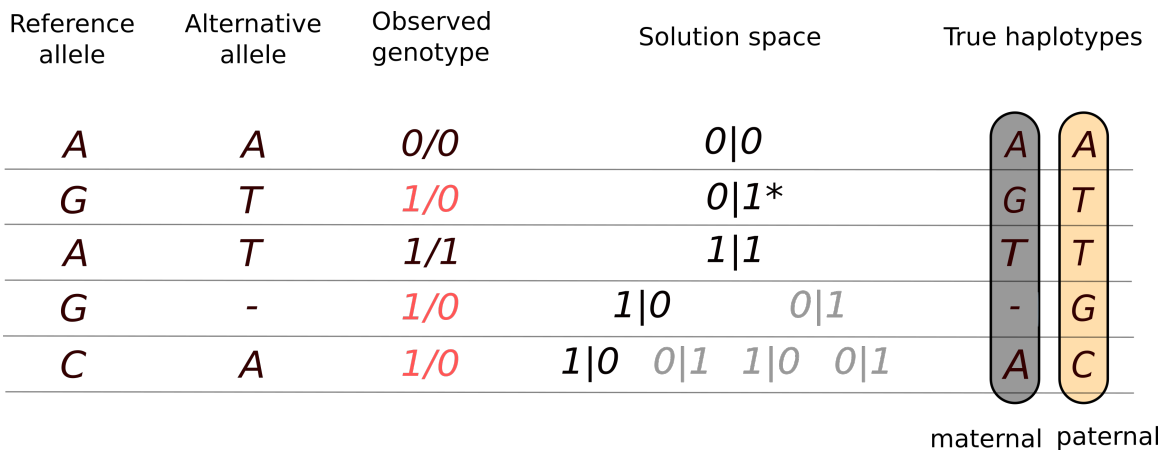


Figure 2.1: Toy example illustrating phasing problem input and output.
(∗) Note that we do not actually know which hapotype came from the father and which from the mother, but we know that each came from a different parents. That is why alleles of the first ambiguous variant is randomly assigned to one of the haplotypes.

ploring within- and across-population haplotype diversity than by focusing on multilocus genotype diversity.

Finally, phase information is greatly used in order to model recombination points and identify variants linkage and interactions among multiple genes and alleles.

2.1.2 Ambiguity and Data sources

To understand the problem, let us consider a toy *.vcf* illustrated in Fig. 2.1. It contains five variants only, three of which are heterozygous (those are highlighted in red). Phase information in those is uncertain and the rest is identical for both estimated haplotypes.

The tree illustrated in Fig. 2.1 represents phasing problem solution space. Here, each path from the tree root to a leaf represent a consistent with the observed genotype solution. Considering a general task of haplotype estimation for n heterogeneous markers, there are 2^{n-1} consistent possibilities. Even so, naturally, only one of the pairs corresponds to the true haplotypes.

There are two additional information sources, allowing to estimate a diplotype correctly and to avoid ambiguity. We refer to the sources as *Reference Panel* and *Read panel*.

Reference panel

Reference panel is an extensive high-quality source of genomic data for multiple samples stored in *.vcf* format. Construction of the reference panel is often made on near-close population, i.e. of European ancestry. This ensures presence of IBD patterns (see Sec. 1.1.3) and increases the patterns' length.

The largest currently available reference panel contains haplotypes of 64,976 humans at 39,235,157 SNPs [Conb]. A very complex and expensive pipeline was followed to obtain the desired high-quality data for so many individuals [Conb]. The results provide a foundation

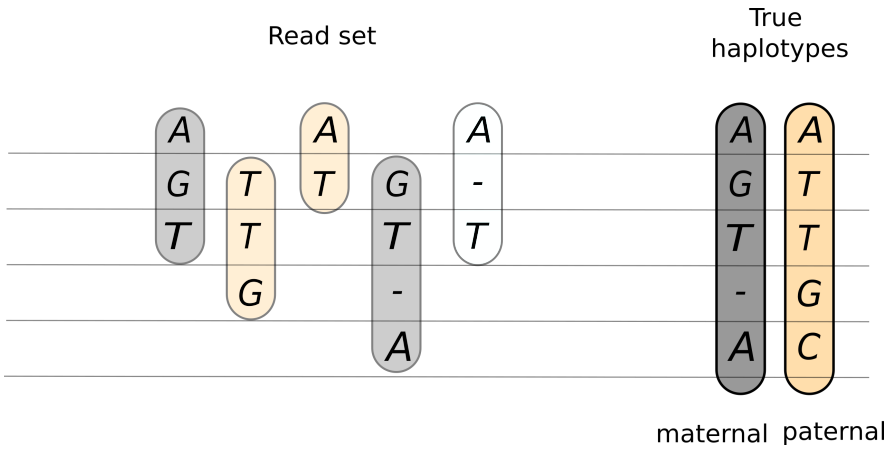


Figure 2.2: Toy example illustrating how phase information is preserved in reads.

for population-based cohort studies such as estimation of distribution, risk factor analysis, recombination model estimation etc. [Szk].

Read panel

Read panel is an abstract term used to refer to the set of reads produced by an individual's genome sequencing and alignment. It is significant that a read panel comprises information about the individual genotype, as well as information about nearby alleles' relation.

Available sequencing techniques, presented in Sec. 1.1.4, ensure that each produced read was entirely sampled from a single haplotype. Therefore, variants which a read covers also have the property.

For instance, let us consider an example introduced previously, but enhanced with aligned reads. Fig. 2.2 illustrates the example. We observe that reads which cover more than one variant allow to partially or completely resolve the ambiguity and to restore the original haplotypes.

Imagine an extreme example, where two reads are present and their lengths are equal to genotype size. Original haplotypes would simply correspond to the reads and phasing would be done in one step.

It was mentioned in Sec. 1.1.4 that no reliable technology able to do molecular sequencing is currently available. Nevertheless, latest sequencing technologies, in particular technologies of the third generation, produce reads of extensive length, covering many consecutive variants. In fact, any read covering more than one variant might be beneficially utilized during phasing.

Genetic map

Another source of information, which might be utilized during phasing, is a genetic map. It contains statistics about propensities towards crossover between chromosome positions measured in centimorgans. Formally, a centimorgan is defined as the distance between chromosome positions for which the expected average number of crossovers in a single generation

is 1%. Roughly speaking, one centimorgan corresponds to about *one* million base pairs in humans on average. However, the relationship varies from place to place in the genome, and also between men and women.

In particular, population-based phasing methods use a genetic map to model recombination event probability distribution.

2.2 Related works

There are plenty of haplotype estimation algorithms, differing in initial assumptions about the input data. Haplotype phasing becomes straightforward if the family genetic information is available or if molecular sequencing is made. However, generally, we do not assume that is the case and rely on other available data sources such as reference panel or read panel, introduced earlier. In general, we also assume that the individual is not directly related to samples in reference panel.

Considering the assumptions listed above, we distinguish between three families of phasing methods, namely population-based phasing, read based-phasing and a hybrid of the two approaches.

2.2.1 Population based phasing

Haplotypes can be inferred from genotype information of large cohorts based on the presence of IBD patterns, which holds for related as well as unrelated individuals as is reviewed in [BBb]. This family of methods is referred to as *population-based phasing* and is based on phased genetic information of a big enough number of individuals, i.e. reference panel. The fact that an unknown haplotype is derived from known haplotypes by mutation and recombination processes is incorporated in many population-based methods, including PHASE [SSD], [SS], its faster implementation SHAPE-IT [DCZ], FastPHASE [SS], Beagle [BBc], Eagle and Eagle2 [LDPea], achieving greater accuracy in comparison with Eagle when reference panel size decreases.

Each of the listed algorithms uses hidden Markov model (HMM) whose parameters are estimated with the use of iterative algorithms such as the stochastic EM algorithm [BBb]. Nevertheless, Eagle2 is capable of achieving high accuracy as well as essential speed-up [LDPea] in comparison with other algorithms even for small reference panels. There are two key ideas distinguishing Eagle2 from existing population-based methods: a new data structure based on the positional Burrows-Wheeler transform [Dur] and a rapid search algorithm that explores only the most relevant phase paths through the HMM [LDPea]. This difference made Eagle2 the state-of-art algorithm for population-based phasing.

Currently available methods for population-based phasing allow to perform haplotype estimation accurately and in acceptable time. However, information present in read panel, which is naturally available after genome sequencing, is not used up by the methods.

2.2.2 Read-based phasing

Recent achievements in sequencing technologies, associated especially with the third generation sequencing (see Sec. 1.1.4), allow reconstructing haplotype from reads' information. As it was discussed in Sec. 2.1.2, read panel provides information that can be incorporated into computational phasing [BBb]. The longer the sequenced fragment, the greater amount of useful information it carries. Building upon the assumptions, *read-based phasing* techniques are being actively developed.

While in 2011 only few methods were available, nowadays there is plenty of algorithms including HapCUT (2008) [BBa], RefHap (2010) [DHM$^+$], Read-backed phasing (2012), H-BOP (2012) [XWJ], ProbHap (2014) [Kul], What'sHap (2015) [PMea] and HapCol (2016) [PZD$^+$]. While being different in methodology, the algorithms also differ in assumptions about the input data, e.g. reads length, sequencing coverage and error rate.

Earlier methods, such as HapCUT, are designed for reads generated with first generation sequencing, mainly Sanger sequencing (low error rate, relatively long reads, $> 700bp$ [HFB]). This method does not outperform currently available methods in terms of accuracy and is slower in general settings [PMea]. Moreover, first generation sequencing is too expensive for whole-genome sequencing on a large scale and is gradually being substituted with the second and the third generations techniques, as it was described in Sec. 1.1.4.

Majority of methods listed above formalize read-based phasing (also called haplotype assembly) as Minimum Error Correction problem (MEC) or its variation weighted MEC (wMEC), where given a set of fragments, the goal is to reconstruct two haplotypes by applying the minimum number of base corrections [BDK$^+$]. The problem is NP-hard [BDK$^+$], but the methods use approximation (RefHap, HapCUT, H-BOP) as well as fix-parameter approaches (What'sHap, HapCol, ProbHap), where the parameter is sequencing depth. Apparently, approximations are faster in general but do not give guarantees on solution quality, while fix-parameter approaches are exact but their time requirements grow exponentially with the growth of the input parameter value.

The latest ones What'sHap and HapCol are mainly designed for the third generation sequencing data. The methods are appropriate for the datasets since What'sHap complexity is not affected by reads' length, while HapCol run time grows only linearly with the length. Although the algorithms run time increases exponentially with increasing read set coverage, authors of What'sHap claim that coverage greater than 20 nearly does not influence result quality, assuming reasonable filtering of input fragments.

All listed methods require some guarantees on input parameters, i.e. sequencing depth, reads' length, error rate, and are not capable to generalize on the input. While third generation sequencing technologies allow retrieving majority of haplotype solely based on reads, they still suffer from high error rate (see Tab. 1.1). Importantly, the performance of the algorithm, primarily designed for the technology, decreases rapidly with increase sequencing error rate [PMea], [PZD$^+$], which makes the methods inefficient for datasets currently produced by the third generation platforms.

2.2.3 Hybrid phasing

Hybrid phasing utilizes availability of both a reference panel and a read panel. The hybridization often allows reaching better algorithm performance in comparing with solo

approaches.

There are several available hybrid algorithms, including SHAPEIT2 (2013) [DHCea], Hap-seq (2013) [HHE] and PhASER (2016) [CMC$^+$]. Hap-seq represents an exact phasing algorithm which is exponential in the number of variants a read covers. Authors fix the number to avoid the exponential complexity, which leads to suboptimal usage of information a read panel supplies. The algorithm is similar to SHAPEIT2, but instead of Markov chain Monte Carlo (MCMC) sampling it uses the exact Viterbi algorithm.

SHAPEIT2 algorithm extends population-based phasing algorithm SHAPEIT by likelihoods computed under the read panel. This extension, among other, allows phasing variants, that are singletons and not present in reference panel [DHCea]. The algorithm, similarly to Hap-seq, iterates by segments of fixed size (from 0.1Mbp to 5Mbp). Differently from the Hap-seq algorithm, the model counts with reads which span the segments' boundaries. The algorithm outperforms SHAPEIT as well as Beagle [BBc] population-based phasing algorithm in terms of phasing accuracy [DHCea]. Accuracy of SHAPEIT2 algorithm increases with increasing reads' length. This property makes the algorithm more suitable for the third generation sequencing technologies. However, it is reported in [DHCea] that when paired-end short reads were supplied (similar to the output of the second generation sequencing platform *Illumina*), increase in the accuracy was observed as well. In particular, for variants that are not in the reference panel, the switch error dropped from 49.3% to 37.1% and at sites with a minor allele count of *eight* the switch error dropped from 2.2% to 1.7%. This underlines the ability of the short reads to increase phasing accuracy.

The newest algorithm, PhASER, is primarily designed for RNA-seq. The algorithm, similarly to other presented hybrid approaches, iterates over blocks (up to 15 variants) and extensively searches the solution space. PhASER constructs a solution space basing on a read panel and utilizes reference panel to compute haplotypes likelihoods. Run time and accuracy of the algorithm for WGS datasets are similar to performance of the HapCUT population-based phasing algorithm [BBa], introduced earlier [CMC$^+$].

To best of authors knowledge, there is only one hybrid algorithm SHAPEIT2, which combines both read panel and reference panel into a single probabilistic model, while fully utilizing phase information contained in reads. It achieves better performance than it's population based phasing counterpart, underlining ability of read panel to improve population-based phasing. Its time complexity is similar to SHAPEIT population-based phasing algorithm.

2.3 Objectives

While currently available read-based phasing algorithms utilize the ability of long reads to span more variants, they are inefficient for short reads produced by the first and the second sequencing technologies, as was reported in e.g. [PMea]. However, high sequencing error rate the long reads suffer from makes the algorithms inefficient for low coverage data. For instance, What's Hap algorithm for PacBio datasets achieves switch error around 10% while phasing both substitutions and INDELs for simulated data and error rate around 15% on real PacBio dataset with coverage 5, as is reported in [PMea]. Increasing the coverage to 60 reduces the errors to 3.8% for simulated dataset and to 8.3% for real dataset. However, in general datasets does not have such depth due to cost of such sequencing and time demands. Moreover, the most efficient read-based phasers (e.g. [PMea], [PZD$^+$], [Kul]) are exponential

in read dataset coverage.

Meanwhile, population-based phasing algorithm are very expensive in term of run time, except Eagle2 algorithm. For instance, in [LDPea] it is reported that Eagle2 achieves 12–38x speedups over SHAPEIT in performance on a reference panels of size $15,000$–$100,000$.

Therefore, the goal of our research is to extend Eagle2 phasing algorithm [LDPea], current the state-of-art population-based phasing algorithm, with information preserved in reads. The information is nearly always available since genome sequencing is the first step performed in any genome analysis pipeline. The presented algorithm aimed at being efficient in term of run time and phasing accuracy for reads obtained with variant available sequencing technologies.

2.4 Student contribution

The student contribution was reviewing the existing phasing methods and design of the Eagle2 algorithm extension with read-based phasing. This includes proposal of the probabilistic model, implementation of the read-based phasing part of the algorithm (including implementation of the tool for reads preprocessing and conversion into the desired format), data simulation and testing of the hybrid algorithm.

2.5 Outline

The thesis is structured as follows. In the previous two chapters, we introduced basic terms, the phasing problem itself and listed currently available phasing algorithms. The following chapter introduces the proposed hybrid phasing algorithm. In the chapter, we formally describe the proposed probabilistic model and its approximation with HMM. In chapter 5 we describe implementation details of the algorithm. The last chapter contains testing results and a discussion about them.

Part II

Materials and Methods

Chapter 3

Eagle2 phasing algorithm

The proposed phasing algorithm performs diplotype inference under the proposed HMM model. The core idea of the method corresponds to Eagle2, which is currently the state-of-art population-based phasing algorithm. The proposed algorithm extends the probabilistic model of Eagle2 by posteriori probabilities computed under a read panel.

In this chapter we briefly describe the Eagle2 algorithm and scope of the proposed extensions. The formal description of the proposed probabilistic model and the haplotype inference procedure, performed under the model, is introduced in the following chapter.

3.1 Algorithm overview

A statistical model of the Eagle2 algorithm is a haplotype copying model similar to the one introduced by Li N. and Stephens M. in 2003 [LS]. The Li and Stephens framework accounts for the biological processes of recombination and mutation to estimate an individual diplotype from known haplotypes of other individuals [BBb], i.e. reference panel. The model forms a base of probabilistic models of the majority of population-based phasing algorithms, e.g. PHASE [SSD], SHEPEIT [DCZ], Beagle [BBc].

However, Eagle2 differs from the dynamic programming or sampling methods employed by previous phasing software. Before haplotype space exploration, the algorithm creates a HapHedge structure, allowing efficiently representing haplotypes from reference panel. Furthermore, Eagle2 selectively explores the diplotype space. We give a brief description of the main algorithm steps.

3.1.1 Step 1: HapHedge data structure construction

A HapHedge structure encodes a sequence of haplotype prefix trees rooted at a sequence of starting positions along the chromosome. This enables a near constant-time retrieval of haplotype frequencies at any position along the chromosome. The property makes phasing algorithm speed nearly independent on reference panel size. Moreover, the structure construction has linear-time complexity and near constant-time querying.

The HapHedge structure is generated using the positional Burrows-Wheeler transform [Dur]. In its simplest form, the PBWT iteratively creates sorted lists of haplotype prefixes,

moving the prefix start point from right to left [LDPea]. Eagle algorithm extends this procedure to convert the lists of sorted prefixes into prefix trees [LDPea]. The procedure takes $\mathcal{O}(M * K)$ time, where K is the number of individuals in reference panel, M is the number of variants to phase. After the structure is created, the algorithm starts to explore the diplotype space.

3.1.2 Step 2: Exploration of diplotype space

The algorithm performs a solution space exploration moving from left to right along the chromosome. At each target marker, the algorithm considers two possible extensions of the haplotype pair as well as possible recombination on maternal, paternal or both haplotypes. State likelihoods are then computed under the reference panel (by querying the HapHedge structure constructed in the previous step) and under the given unphased genotype of the sample.

States merging

Apparently, the number of haplotypes in the reference panel which are consistent with the haplotype will continually decrease when moving from left to right along the chromosome. This reduces the ability of the algorithm to overcome local maxima. The behavior is prevented so that states, for which haplotypes agree in the last 64 heterozygous positions, are merged into a single state and the new state likelihood is approximated by a weighted combination of the merged states likelihoods (for more details see [LDPea] Sec. 2.3.2). The step also reduces the number of states, resulting in a more tractable HMM.

Space pruning

To prevent the exponential explosion of the solution space, along with states merging, space pruning is performed. The solution space is pruned to the P most likely solutions in each step. The hope is that P is set to be big enough to be robust to local maxima [LDPea].

Therefore, the algorithm performs beam search of a small subset of the solution space, which is advantageous when most of the space is probabilistically unfavorable but difficult to discard a priori [LDPea].

3.2 Proposed extension

Newly, we add read set posteriori probabilities to the Eagle2 probabilistic model. In the proposed algorithm, we construct an efficient reference panel representation (different from HapHedge structure) using PBWT at first. Then, we precompute a posteriori probability of each read alignment contained in a supplied *.bam* file, given alleles from a supplied *.vcf* file. Further, the probabilities are incorporated into the probabilistic model, which takes into account both sources of information: the created read panel and the supplied reference panel. Then we, similarly to Eagle2, approximate the proposed model with HMM and iterate over it using Viterbi Algorithm, performing states merging and pruning at each step of the inference. The proposed probabilistic model and the inference procedure are formally described in detail in the following chapter.

Chapter 4

Hybrid phasing algorithm

The following chapter contains a formal description of the proposed probabilistic model as well as a diplotype inference performed under the model.

4.1 Preliminaries

The following hybrid algorithm is based on two sources of information, read panel R and reference panel H. Let N be the total number of variants (or markers) in a considered region of a genome. We further refer to reads, covering a position p as R_p, where $p \in < 1, N >$.

We represent a diplotype as a pair of maternal and paternal haplotypes denoted as h^α and h^β respectively. We refer to the value of a haplotype at a specific position $p \in < 1, N >$ as h_p. We consider both heterozygous and homozygous variants since homozygous are important for inference under H.

4.2 Diplotype probability model

Our goal is to model a distribution of all possible diplotypes, given the available information from reads and taking into account population haplotype structure, i.e. $\mathbf{p}(h^\alpha, h^\beta | R, H)$. We also take into account recombinations and a possibility of discrepancies between the sources.

Assuming independence of R and H given a pair of haplotypes h^α, h^β (similarly to [DHCea]) we rewrite the probability $\mathbf{p}(h^\alpha, h^\beta | R, H)$ as follows

$$\begin{gathered} \mathbf{p}(h^\alpha, h^\beta | R, H) \propto \mathbf{p}(h^\alpha, h^\beta, R, H) = \\ \mathbf{p}(H, h^\alpha, h^\beta) \cdot \mathbf{p}(R | h^\alpha, h^\beta, H) \propto \\ \mathbf{p}(h^\alpha, h^\beta | H) \cdot \mathbf{p}(R | h^\alpha, h^\beta) \end{gathered} \tag{4.1}$$

This allows us to divide our model into two independent parts, referred to as the *haplotype model* and the *read model*.

4.2.1 Haplotype probability model

We first approximate conditional diplotype distribution given H by making independence assumption on maternal and paternal haplotypes given H (similarly to [LDPea]), where the haplotypes are coming from the same distribution on $h_{1:N}$, i.e.

$$\mathbf{p}(h^{\alpha}_{1:N}, h^{\beta}_{1:N}|H) \approx \mathbf{p}(h^{\alpha}_{1:N}|H)\mathbf{p}(h^{\beta}_{1:N}|H) \tag{4.2}$$

Next, we model each haplotype as a mosaic of recombined haplotype segments from $h_{1:a}, h_{a+1:b}, \dots h_{l+1:N}$, copied from haplotypes from H. Here, we denote a set of recombination breakpoints for a haplotype as $a, b, \dots l$, and it holds that $1 < a < b < c \dots l < N$. We can thus decompose a probability of haplotype as a sum over the possible recombination breakpoints as

$$\begin{aligned} \mathbf{p}(h_{1:M}|H) = \sum_{a,b,\dots l} \mathbf{f}(h_{1:a}|H)\mathbf{p}_{rec}(a) \\ \mathbf{f}(h_{a+1:b}|H)\mathbf{p}_{rec}(b) \dots \mathbf{f}(h_{l+1:M}|H)\mathbf{p}_{rec}(M), \end{aligned} \tag{4.3}$$

where $\mathbf{p}_{rec}(p), p \in <1, N>$ models the probability of a breakpoint in position p and $\mathbf{f}(h_{a:b}|H)$ denotes the frequency of a haplotype segment $h_{a:b}$ in H, i.e.

$$\mathbf{f}(h_{a:b}|H) = \frac{\{k : h^{k}_{a:b} == h_{a:b}\}}{K}, \tag{4.4}$$

where K denotes the reference panel size, i.e. $K = |H|$.

Recombination probability model

We use Li-Stephens model [LS] to describe probability $\mathbf{p}_{rec}(p), p \in <1, M>$ that a recombination event occurs at the marker p. Under this model the probability is computed as

$$\mathbf{p}_{rec}(p) = 1 - e^{-\lambda x} = 1 - e^{-\lambda(G(p) - G(p_{prev}))}, \tag{4.5}$$

where λ is the expected IBD length and $G(p)$ denotes the genetic distance of marker p from the beginning of the chromosome, which can be found in genetic map (see Sec. 2.1.2 for details). The parameter λ corresponds to theoretical distance of a phased sample to samples from the reference panel H. And as it was shown in [LDPea], phasing accuracy appears to be insensitive to λ from the range $0.5cM - 4cM$.

4.2.2 Read model

Similarly to SHAPEIT2 [DHCea], we compute the posteriori probability of R given a pair of haplotypes as follows

$$\mathbf{p}(R|h^{\alpha}, h^{\beta}) = \prod_{r \in R} \mathbf{p}(r|h^{\alpha}, h^{\beta}) \tag{4.6}$$

Assuming that prior probabilities of the haplotypes being sequenced are equal, we rewrite the equation as follows

$$\prod_{r \in R} \mathbf{p}(r|h^{\alpha}, h^{\beta}) = \prod_{r \in R} \frac{\mathbf{p}(r|h^{\alpha}) + \mathbf{p}(r|h^{\beta})}{2} \tag{4.7}$$

Assuming independence of bases within a read, we utilize read base qualities, available after sequencing to rewrite posteriori of a read as follows

$$\mathbf{p}(r|h) = \prod_{i=1}^{|r|} \mathbf{p}(r_i|h) = \begin{cases} q(r_i) & r_i = h_{p(r_i)} \\ 1 - q(r_i) & r_i \neq h_{p(r_i)} \end{cases} \tag{4.8}$$

where $q(r_i)$ is a probability of the base r_i being sequenced correctly and $p(r_i)$ is position of the base.

4.3 Haplotype inference

It turns out that each multiplier in the haplotype model (4.3) can be computed in $\mathcal{O}(1)$ time. For the recombination model introduced earlier, the complexity is straightforward. Using the PBWT algorithm [Dur], we efficiently represent H as a structure which supports lookup of the fraction value (4.4) in constant time. Therefore, it remains to compute the exponential sum over all possible recombination breakpoints and over possible diplotypes. We approximate the sum by the following Hidden Markov model (HMM).

4.3.1 Hidden Markov model

We denote a state of HMM as S, while S^{-1} denotes an antecedent state of S. A step further is always made upon a single marker p, i.e. $h^{\alpha}(S) = h^{\alpha}(S^{-1}) + h^{\alpha}_{p}$, where $h^{\alpha}_{p} \in \{0, 1\}$. Here *zero* denotes consensus with the reference, while *one* denotes a consensus with the alternative allele (see Sec. 2.1 for details). Thus, a state S represents

- a position $p \in <1, N>$ along the chromosome.
- e^{α} and e^{β}, denoting positions where the latest recombination event occurs for maternal and paternal haplotype respectively, thus $e^{\alpha} \in <1, p>$ as well as $e^{\beta} \in <1, p>$.
- h^{α} and h^{β}, denoting maternal and paternal haplotype sequences up to the current position p, i.e. the sequences begin in the first position and continue up the current position p.

4.3.2 State space exploration

We move from left to right along the chromosome. Being in position p, we consider both possible diplotypes, i.e. $h^{\alpha}_{p+1} = 1$, while $h^{\beta}_{p+1} = 0$ and $h^{\alpha}_{p+1} = 0$, while $h^{\beta}_{p+1} = 1$ for each heterozygous variant. For homozygous variants, we consider a single case where haplotypes are equal. We also consider the possibility of a recombination in p on maternal, paternal, or

both haplotypes.
Probability of the state S given S^{-1} is computed as follows

$$\mathbf{p}(S|H,R,S^{-1}) = \mathbf{p}(S|H,S^{-1})\mathbf{p}(S|R,S^{-1}), \tag{4.9}$$

which follows from (4.1). Then, $\mathbf{p}(S|H,S^{-1})$ is computed as

$$\mathbf{p}(S|H,S^{-1}) = \frac{\mathbf{p}(S,H,S^{-1})}{\mathbf{p}(H,S^{-1})} = \frac{\mathbf{p}(S,S^{-1}|H)\mathbf{p}(H)}{\mathbf{p}(S^{-1}|H)\mathbf{p}(H)} = \frac{\mathbf{p}(S,S^{-1}|H)}{\mathbf{p}(S^{-1}|H)} \tag{4.10}$$

The second part of the (4.9), $\mathbf{p}(S|R,S^{-1})$, is computed as

$$\mathbf{p}(S|R,S^{-1}) = \frac{\mathbf{p}(S,S^{-1},R)}{\mathbf{p}(S^{-1},R)} = \frac{\mathbf{p}(S,S^{-1}|R)\mathbf{p}(R)}{\mathbf{p}(S^{-1}|R)\mathbf{p}(R)} = \frac{\mathbf{p}(S,S^{-1}|R)}{\mathbf{p}(S^{-1}|R)} \tag{4.11}$$

Further, we decompose R into $R_{:p(S^{-1})}$, $R_{p(S):}$ and $R_{:p(S):}$ i.e. reads, which end in positions $p(S^{-1})$, which start at position $p(S)$ and which span both respectively. Reads which do not belong to any of the sets do not influence (4.6) according to (4.8). This allows as to rewrite the equation above as

$$\begin{gathered}
\frac{\mathbf{p}(S,S^{-1}|R)}{\mathbf{p}(S^{-1}|R)} \propto \frac{\mathbf{p}(R|S,S^{-1})}{\mathbf{p}(R|S^{-1})} = \\
\frac{\mathbf{p}(R_{:p(S^{-1})},R_{p(S):},R_{:p(S):}|S,S^{-1})}{\mathbf{p}(R_{:p(S^{-1})},R_{p(S):},R_{:p(S):}|S^{-1})} = \\
\frac{\mathbf{p}(R_{:p(S):}|S,S^{-1})\mathbf{p}(R_{:p(S^{-1})}|S^{-1})\mathbf{p}(R_{p(S):}|S)}{\mathbf{p}(R_{:p(S^{-1})}|S^{-1})\mathbf{p}(R_{:p(S):}|S^{-1})} = \\
\frac{\mathbf{p}(R_{:p(S):}|S,S^{-1})\mathbf{p}(R_{p(S):}|S)}{\mathbf{p}(R_{:p(S):}|S^{-1})}
\end{gathered} \tag{4.12}$$

In case of a recombination for state S on haplotype h the so called *states merging* occur. The step is necessary to preserve beam diversities, as it was already mentioned in Sec. 3.1.2. In this case we merge states which correspond to the previous position and which agree in a considered haplotype h from the previous recombination point e to the current position $p(S)$. More formally, states from the state set $\mathcal{S}^M = \{S^m | p(S^m) = p(S^{-1})$ *and* $\forall S_1^m, S_2^m \in \mathcal{S}^M : h_{e:}(S_1^m) = h_{e:}(S_2^m)\}$ are merged into a single state which we refer to as S'.

Therefore, the likelihood of S' becomes conditioned on likelihoods of the states from $\mathcal{S}^M$. We approximate the probability as follows

$$\mathbf{p}(S'|H,R,S^M) = \sum_{S^m \in \mathcal{S}^M} \mathbf{p}(S^m|H,R) \tag{4.13}$$

In the case of states merging, we also approximate the posteriori $\mathbf{p}(r|h(S'))$ for haplotype h by the weighted sum of the read r posteriors over states from $\mathcal{S}^M$ as follows

$$\mathbf{p}(r|h(S')) = \frac{\sum_{S^m \in \mathcal{S}^M} \mathbf{p}(r|h(S^m))\mathbf{p}(S^m|H,R)}{\sum_{S^m \in \mathcal{S}^M} \mathbf{p}(S^m|H,R)} \tag{4.14}$$

Chapter 5

Implementation

While the high-level implementation of the proposed method is similar to the Eagle2, the algorithms differ in several ways. First of all, we use a different representation of known haplotypes from H we refer to as *PBWT Index*. Next, we incorporate probabilities of reads from R into the model. A diplotype inference under the proposed HMM remains the same for both algorithms.

Therefore, the proposed algorithm could be divided into three steps

- **Step 1**: Construct a read panel.
- **Step 2**: Build *Index* using the PBWT algorithm.
- **Step 3**: Infer the most likely diplotype under the model introduced in Sec. 4.

We had implemented a new bioinformatics tool which creates a read panel structure in a desired format. Since the structure might be efficiently utilized for other application, the *step 1* is implemented separately from the entire phasing algorithm.

In this chapter we first describe implementation details of the algorithm performing a read panel construction. Then we describe implementation of the hybrid phasing algorithm and discuss those algorithms' time complexity. Also, we list the algorithms' requirements, input arguments, and present examples of execution commands in the chapter.

5.1 Tool for Read Panel Construction

The implemented tool for read panel creation produces a *.var* file, given *.vcf* and *.bam* files. The newly proposed file *.var* format is designed in a way allowing to iterate over variants listed in a *.vcf* while preserving information about consecutive variants being spanned by a single read.

Briefly, the tool main cycle iterates over reads in the given sorted *.bam* file and as soon as a read covers a variant from the given *.vcf*, it accesses the read alignment to the variant and calculates the aligned bases probability for each allele of the variant according to (4.8). While iterating over the read set, the algorithm simultaneously moves along the sorted variant set. To reduce the memory overhead, results are frequently stored into a file during the

algorithm runtime.

The algorithm is capable of processing all types of reads (including paired-end reads produced by the Illimina platform) of any length and base qualities. The algorithm time complexity is $\mathcal{O}(|R| * v_{max})$, where $|R|$ is number of aligned reads and v_{max} is the maximum number of variants a read from R spans.

5.1.1 The .var format

```
2__err_0.01-chr20_haplotype_2-17015 5714929:5715284:5718030 0,8:0,7:2,4
2__err_0.01-chr20_haplotype_2-16544 5714929:5715284:5718030 2,4:0,13:0,13
1__err_0.01-chr20_haplotype_1-17908 5714929:5715284:5718030:5719739 12,0:2,4:12,0:11,0
1__err_0.01-chr20_haplotype_1-18013 5715284:5718030:5719739 7,0:13,0:10,0
1__err_0.01-chr20_haplotype_1-17012 5718030:5719739:5720394 6,1:2,4:8,0
2__err_0.01-chr20_haplotype_2-17632 5718030:5719739:5720394 0,12:4,2:4,2
1__err_0.01-chr20_haplotype_1-17916 5718030:5719739:5720394:5721181 11,0:12,0:1,6:4,16
1__err_0.01-chr20_haplotype_1-16763 5718030:5719739:5720394:5721181 3,3:10,0:3,3:0,7
1__err_0.01-chr20_haplotype_1-17668 5718030:5719739:5720394:5721181 3,3:3,3:9,0:9,5
```

Figure 5.1: Example of the *.var* format.

The *.var* file, produced by the tool, contains the posteriori of reads given variant alleles as well as reads' base qualities calculated according to (4.8). Example of a file in *.var* format is presented in Fig. 5.1. Each line of the file contains information about a single read and is composed by the read name, the list of covered markers' start positions separated by semicolons, and the list of the variants posteriori separated by semicolons. Each variant posteriori is in form of a list where each element is a comma separated list of posteriori of the variant alleles.

The variant posteriori probability is presented in a format common for base quality representation, known as *Phred* or *Q score*. A Q score value is an integer value representing the estimated probability of an error, i.e. that the base is incorrect. The representation helps to avoid manipulations with floating point numbers. If p is the error probability and q is a Q score, then conversion between the formats is performed as:

$$\begin{aligned} p &= 10^{-\frac{q}{10}} \\ q &= -10log_{10}(p) \end{aligned} \tag{5.1}$$

The variant alleles posteriori probability are computed from sequencing qualities of the aligned read bases, which are available in a *.bam* file in the Phread format. Therefore, the probability for a read can be computed as long as the read cigar string (see Sec. 1.2), containing information about the read alignment, is available in the *.bam* file.

5.1.2 Execution

Dependencies

The tool requires *Python 3.x* to be installed. Also, it requires *numpy*, *pysam* and *biopython* packages to be installed.

Arguments

The script arguments are

- Positional arguments

```
bam_path - absolute path to the .bam file
vcf_path - absolute path to the .vcf file
out_prefix - prefix of the output .var file, including file location
sample_id - sample id, identical to the id in the .vcf file
```

- Optional arguments

```
--config - configuration/reference id specified in the .vcf file
--varmax - the position at which variant processing will end [All]
--homo - include variants in which the sample is homozygous [False]
--indels - include INDELs [True]
--v - verbose mode [False]
```

Note that `--config` argument is required to be specified in the case when the given *.vcf* file contains data for multiple reference sequences. The program will set the reference id automatically otherwise.

Examples

Example 1: To create `hg00096_read_panel.var` file for a sample `HG00096` which would contain all SNPs specified in `variants.vcf` (containing data for a single configuration) in which the sample is heterozygous for reads in `reads.bam`, run

```
python3 createVar.py reads.bam
                     variants.vcf
                     hg00096_read_panel
                     HG00096
```

Example 2: To create `hg00096_read_panel.var` file for a sample `HG00096` which would contain substitutions listed in `variants.vcf` (containing data for a single configuration) including variants in which the sample is homozygous, for reads in `reads.bam` run

```
python3 createVar.py reads.bam
                     variants.vcf
                     hg00096_read_panel
                     HG00096
                     --homo=True
                     --indels=False
```

5.2 Phasing Algorithm

5.2.1 Step 1: Read panel construction

As the first step, we apply the tool, presented in the previous section, to access reads posteriori probabilities. We call `createVar.py` for the given *.bam* and *.vcf* containing the sample variants which we want to phase, for heterozygous variants only, for both substitutions and INDELs. We exclude homozygous variants since for phasing under the read panel this variants are invaluable. The phasing algorithm then takes the created *.var* as an input.

5.2.2 Step 2: PBWT Index construction

Next, we build the *Index* structure using the PBWT algorithm [Dur]. The algorithm takes H, containing known haplotypes, and iteratively creates sorted lists of haplotype prefixes, moving the prefix start point from right to left. Therefore, at each position p we poses information about how many individuals have *zero* at positions $p + 1$ and how many have *one*. The information is crucial for inference under the haplotype model (4.2.1).

The index construction takes $\mathcal{O}(N * K)$ time where K is the number of individuals in reference panel and N is total number of markers.

5.2.3 Step 3: Haplotype inference

After both *Index* and read panel structures are constructed, the inference procedure starts. To find the most likely sequence of hidden states, we iterate over the solution space consecutively moving from marker to marker extending each state by *eight* possible scenarios in each heterozygous variant and by *four* scenarios in each other variant.

The branching factors are given by the fact that we consider $four$ possible recombination scenarios, which are no recombination, recombinations on maternal or paternal haplotype only, or recombination on both haplotypes and possible extentions of the haplotype. Being in position p, we consider both possible diplotypes, i.e. $h^{\alpha}_{p+1} = 1$, while $h^{\beta}_{p+1} = 0$ and $h^{\alpha}_{p+1} = 0$, while $h^{\beta}_{p+1} = 1$ for each heterozygous variant and we consider a single case where haplotypes are equal for each homozygous variant. We compute the likelihood of the extended state according to model described in Sec. 4.3.1.

Once we have crossovered states, we merge those with the same history (haplotype till the last recombination breakpoint) into a single state, as it is formalized in Sec. 4.3.2. The likelihood of a state produced by the merging operation is then computed by (4.13), while posteriori of each read is approximated according to (4.14). The merging is performed in such a way that states are first sorted according to the 64 bit hashcode of the state history, after which states with the same hashcodes are merged. After the merging, we extend each state by possible combination(s) of alleles, as described above.

After that, we prune the set by selecting only the P most likely states in order to avoid the exponential growth of the solution space. Therefore, we have a maximum of P states in each step of the proposed HMM. Pseudocode of the complete inference procedure is formalized in Alg. 1.

After this step, we obtain h^{α}_{best}, h^{β}_{best} - the most likely haplotypes. The algorithm then writes all variants in the original $.vcf$ into a new phased $.vcf$.

Data: H - Index, R - Read Panel, g - sample genotype, P
Result: h^{α}_{best}, h^{β}_{best}

$m = 1$
$\mathcal{S} = \{State(h^{\alpha} = "", e^{\alpha} = 0,$ // init state space
$h^{\beta} = "", e^{\beta} = 0)\}$
while $m < N$ **do**
  $\mathcal{S}.update\_references(H)$ // move along Index for $\forall s \in S$
  $\mathcal{S}_m = \mathcal{S}$
  $\mathcal{S}_m.crossover()$ // crossover each s in S_m
  $\mathcal{S}_m.merge()$ // merge states with the same history
  if $g^1_m = g^2_m$ **then**
    $\mathcal{S} = \mathcal{S}_m.extend(g^1_m, g^2_m)$ // extend both h^{α} and h^{β} by $g^1_m = g^2_m$ for $\forall s \in S$
  else
    $\mathcal{S} = \mathcal{S}_m.extend(g^1_m, g^2_m) \cup$ // extend h^{α} by g^1_m and h^{β} by g^2_m for $\forall s \in S$
    $\mathcal{S}_m.extend(g^2_m, g^1_m)$ // extend h^{α} by g^1_m and h^{β} by g^2_m for $\forall s \in S$
  end
  $\mathcal{S}.compute\_likelihoods()$
  $\mathcal{S} = \mathcal{S}.get\_best\_states(P)$ // get P the most likely states
  m += 1
end
$S_{best} = \mathcal{S}.get\_best\_state()$ // get state with the highest likelihood
return $h^{\alpha}(S_{best})$, $h^{\beta}(S_{best})$ // return estimated diplotype

Algorithm 1: High level implementation of the hybrid phasing algorithm.

5.3 Computational cost

In this section we discuss complexity of each step of the algorithm described in the previous section (see. Alg. 1). The algorithm time complexity is primarily determined by the number of variants N, over which the algorithm iterates.

Crossover of a state (notated as *crossover()* in Alg.1) is considered at each variant. It results in four possible scenarios, which are *(1)* no recombination, *(2)* recombination on paternal haplotype, *(3)* recombination of paternal haplotype or *(4)* both. When we consider crossover for a haplotype, we literally forget the state history and rewrite the last crossover point for the haplotype. This operation has near constant time and is performed for each of the P states obtained in the previous step of HMM. Hence, it takes $\mathcal{O}(P)$ time. Evidently, the step extends the state space by a factor of *four*. Since we consider only the most likely P states at each step, the crossover operation results in $4P$ states.

Then we perform states merging (notated as *merge()* in Alg. 1), which requires first to sort all haplotypes of the $4P$ states. The sorting, performed on maternal and on paternal haplotypes histories, requires $\mathcal{O}(2 * 4P * log4P) = \mathcal{O}(P * logP)$. In case of merging, we also perform summation of the merged states' likelihoods together with reads posteriori averaging. This takes $\mathcal{O}(4 * P + 4 * P * C_{max}) = \mathcal{O}(P * C_{max})$ in total.

It has to be mentioned that states merging reduces the number of states $4P$, obtained

during the crossover step.We discuss the reduction in the next section. Nevertheless, we assume that the number of states is $\mathcal{O}(4P)$, as was inferred above, for simplicity .

Next, we extend each state according to *one* (homozygous marker) or *two* (heterozygous marker) possible scenarios, as was described in the previous section. Thus, we result in $\mathcal{O}(8P)$ states after the step. For each such state we compute its likelihood under model described in Sec. 4.3.2. The first term in (4.1) requires $\mathcal{O}(1)$, due to the fact that querying haplotype frequencies under *PBWT Index* structure takes near constant time. Computing of (4.11) value requires to iterate over all reads that span the position, which takes $\mathcal{O}(C_{max})$ time. Thus, the step requires $\mathcal{O}(8 * P * C_{max})$ in total.

Next, we select P the best states according to their likelihoods at each step in $\mathcal{O}(8 * P * log8P) = \mathcal{O}(P * logP)$ time, due to sorting of the states by their likelihoods. Those selected states are then propagated to the next iteration of the HMM.

Overall, the phasing algorithm computational complexity is $\mathcal{O}(N * P(LogP + C_{max}))$.

5.3.1 Discussion

It was shown in the previous section that the overall complexity of the algorithm is linearly depend on number the of markers N as well as on the number of HMM states P we remember in each iteration. Regarding the remaining summation, we could not say for sure which of P and C_{max} would be dominated. Those values depends on user preferences and on input data. In general settings we assume that $P \in < 100, 1000 >$ and $C_{max} \in < 5, 30 >$. However, it does not hold universally. For example, deep sequencing, which is used to identify mutations within tumors or other rare clonal types, or microbes comprising as little as 1% of the original sample [Ill], might have depth ranging from 80× up to *thousands*.

Hyperparameter *P*

The only hyperparameter of the algorithm is the number of states P we remember in each step of HMM. While the time requirements of the algorithm would grow exponentially with unbounded P, we expect that reasonable settings of P will be a good approximation. We had also expected that accuracy of phasing would grow with increasing P, since unbounded number of states would lead to an exact solution under the proposed model. However, we had empirically estimated that the relation is not straightforward and results obtained for smaller P values are comparable to those obtained with higher P values. We discuss setting of the parameter in the following chapter.

States merging

As is was already mentioned in the previous section, the number of states in often essentially reduced after states merging operation is performed. Unfortunately, the factor could not be exactly inferred and has to be estimated empirically. This is because results of the merging is highly influenced by data present in reference panel as well as by decisions made in the previous steps of the inference procedure.

5.4 Execution

The hybrid phasing algorithm is implemented in *Rust* programming language. To execute the program user runs `phase.exe` located in `target/release/` directory. Along with other arguments listed below, the program takes a *.vcf* containing unphased genotype information and outputs a *.vcf* with the phased genotype.

Arguments

The program arguments are

- Required arguments

```
-p, --panel - absolute path to the reference panel (.vcf/.bcf)
-i, --input - absolute path to variation sample (.vcf/.bcf)
-o, --output - prefix of the output .vcf file, including file location
-g, --gmap - absolute path to the genetic map file
```

- Optional arguments

```
-r, --reads - read panel file name (.var) [None]
-M, --markers - # markers to phase [All]
-N, --refs - # haplotypes from the reference panel which will be used [All]
-D, --drop-states - # HMM states [100]
```

Since the algorithm can phase also without reads panel, `--reads` argument is optional. Also, if `--input` argument specifies a *.vcf* containing more than one sample, only the first one will be phased.

Examples

To run the program with a read panel, a *.var* file has to be created first using the tool introduced in Sec. 5.1. The file is then supplied to the program using `--reads` argument.

Example 1: To phase `hg00096_unphased.vcf` into `hg00096_phased.vcf` in a short time, utilizing all haplotypes from the reference panel `reference_panel.vcf` and a read panel `hg00096_read_panel.var`, run

```
phase -p reference_panel.vcf
      -i hg00096_unphased.vcf
      -o hg00096_phased
      -r hg00096_read_panel.var
      -g gmap.txt
      -D 100
```

This command produces a `hg00096_phased-best.vcf.gz`, containing the most likely diplotype estimated for the sample. Note that `--D` parameter corresponding to number of HMM states remembered at each iteration influences the algorithm's computational time.

Example 2: To phase the first 1000 variants from `hg00096_unphased.vcf` into `hg00096_phased.vcf` in a short time, utilizing 500 haplotypes from the reference panel `reference_panel.vcf` and a read panel `hg00096_read_panel.var`, run

```
phase -p reference_panel.vcf
      -i hg00096_unphased.vcf
      -o hg00096_phased
      -r hg00096_read_panel.var
      -g gmap.txt
      -D 100
      -M 1000
      -N 500
```

Part III

Results

Chapter 6

Testing design

6.1 Testing scenarios

We were interested in how different reads' parameters influence the phasing algorithm performance. In order to examine that, we proposed our testing pipeline as follows

- First of all, we simulated reads of lengths $1k$, $5k$, and $10k$ *bp* with uniform sequencing errors of 1%, 5% and 10% (from which 10% are always INDELs) for 20 samples. In this scenario, average sequencing depth was 15 and P, the number of HMM states, was 100. We aimed at examining how reads' parameters influence the algorithm performance. We also compared the obtained results with those measured without a read panel.
- We tested the algorithm on the dataset described above for $P = 1000$. Our goal was to compare the algorithm performance for number of HMM states $P = 100$ and $P = 1000$.
- We simulated a dataset containing reads of length $10k$ *bp* and sequencing error rate of 20%. We were interested in verifying the algorithm ability to improve phasing accuracy even when given reads with extremely high sequencing error rate.
- We reduced average dataset coverage from 15 to 5 reads of length $10k$ and sequencing error rates of 10% and 20% to test the algorithm performance on low coverage data with high errors rates (such as reads produced with the third generation sequencing platforms).

6.2 Data

All datasets were simulated and tested on the first $475k$ markers which constitute $1/4$ of the human *chromosome 20*. The chromosome in total contains approximatively 64 million base pairs and 1.8 million markers.

6.2.1 Data sources

We used a phased *.vcf* containing approximately 2400 samples which were announced by *phase3* of the *1000 Genome project* [Cona]. We phased the first 20 samples available in the file and the rest was used as a reference panel.

6.2.2 Dataset simulation

We applied our algorithm to reads created with the *NEAT-genReads* sequining simulator [SHM+]. We learned a base qualities model on a real PacBio dataset first, then during the simulation we rescaled the model to obtain the desired error rate and reads length.

In order to simulate reads for a sample, we first created a *.fa* file (see Sec. 1.2), containing genome sequence of the sample for each haplotype separately. This was accomplished by altering variants from a phased *.vcf*, containing the sample information, into a reference sequence. Then we applied the *NEAT-genReads* tool with the pretrained base error model to each of the two altered reference sequences. Once both *.fq* files were created, we merge them into a single file and apply *bwa* aligner tool [LD] in order to obtain the *.bam* file. This file was then used to create a *.var* file, an input of the phasing algorithm.

6.3 Performance evaluation

To evaluate the phasing algorithm performance we used a common metric called *Switch Error*, which is the percentage of possible switches in haplotype orientation, used to recover the correct phase in an individual [LCC].

In order to get the error rate for two given *.vcf* files, the first one containing the ground truth and the second containing the estimated phase information, we used *gzdiff* option of *vcftools* software [DAAea].

6.3.1 Environment

Testing was conducted on a machine with 32 GB of RAM and the Intel Core $i7-4790$ CPU (clocked at 3.6 GHz) with 8 logical cores. *Nine* threads were always executed in parallel which might have influenced the tested algorithms' runtime. All the files had been stored on an external HDD.

6.4 Number of HMM states

We first tested the algorithm for a randomly selected sample to select a suitable P. We measure the performance for a single sample for various reads parameters (on the dataset we constructed for the first testing scenario). We varied P in the set $\{100, 200, 300, 500, 700, 1000\}$.

The switch error rate we measured for sample $HG00100$ for IBD length equal to $1cM$ is illustrated in Fig. 6.1 via boxplots for each value of P. We had noticed that error rate is not straightly proportional to P. No relation between P and the algorithm accuracy had been observed in the results of the experiment. However, from the box plot on the right side of Fig. 6.1, illustrating the phasing runtime, we observe a significant increase with growing P (approximately linear, as was discussed in Sec. 5.3).

From the results of the experiment we concluded that for the selected IBD length accuracy of the algorithm does not vary significantly for the tested P values. Therefore, we decided to test the algorithm for $P = 100$ and $P = 1000$. The hope is that for $P = 100$ the

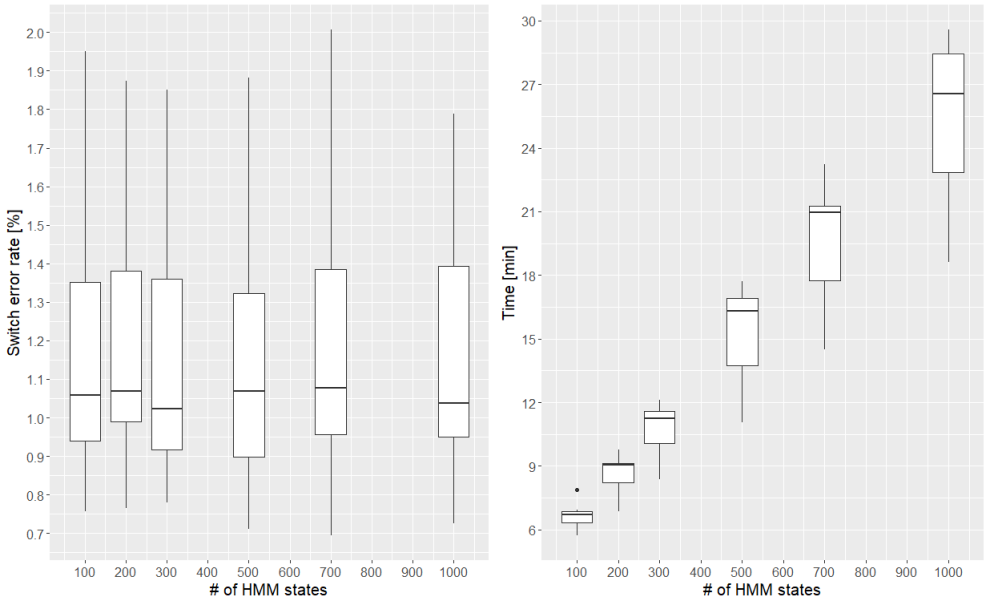


Figure 6.1: Box plots of the relation between the number of HMM states and the switch error rate and time, respectively.

algorithm will perform significantly faster, but for $P = 1000$ we expect phasing accuracy to be slightly higher at the expense of computational efficiency.

Chapter 7

Results and discussion

7.1 Results

7.1.1 Scenario 1: Impact of read panel

First of all, we tested algorithm performance for various read lengths and error rates for average dataset of coverage 15 and $P = 100$. The obtained switch error rates are illustrated in the left side plot in Fig. 7.1. A gray box represents results of phaser with no reads, colorful box-plots represent results of phasing with reads of various parameters. We observe that the algorithm performed better with presence of reads. Importantly, it holds also for reads having sequencing error rate of 10%.

Specifically, the algorithm achieved 1.4% mean switch error over 20 phased samples for phasing without reads, while for phasing with reads having 10% sequencing error rate, it achieved error of 1.04%, 0.68% and 0.53% for reads of length $1k$, $5k$ and $10k$, respectively. The lowest achieved mean switch error was 0.37% for reads of length $10k$ *bp* and sequencing error rate of 1%, as was expected.

From the right side plot of the Fig. 7.1, illustrating CPU time, we observe that time requirements of the algorithm for phasing with reads was slightly higher. It averaged at 6.4 minutes for phasing without a read panel, and 6.6, 7.5 and 7.7 minutes for phasing with reads having 1% sequencing error rate and length equal to $1k$, $5k$ and $10k$, respectively. The figure underlines that the run time is independent on sequencing error rate and near-independent on reads length.

7.1.2 Scenario 2: Number of HMM states

We increased P to 1000 and applied the phasing algorithm to the data used in the previous scenario. Fig. 7.2 illustrates the achieved error rates for $P = 100$ and $P = 1000$. We can see that the accuracy achieved for various reads length and sequencing error rates is practically independent on P, which is consistent with the results reported earlier in Sec. 6.4 measured for a single sample.

The presented results were obtained for IBD length equal to $1cM$ and could vary for different values of the parameter.

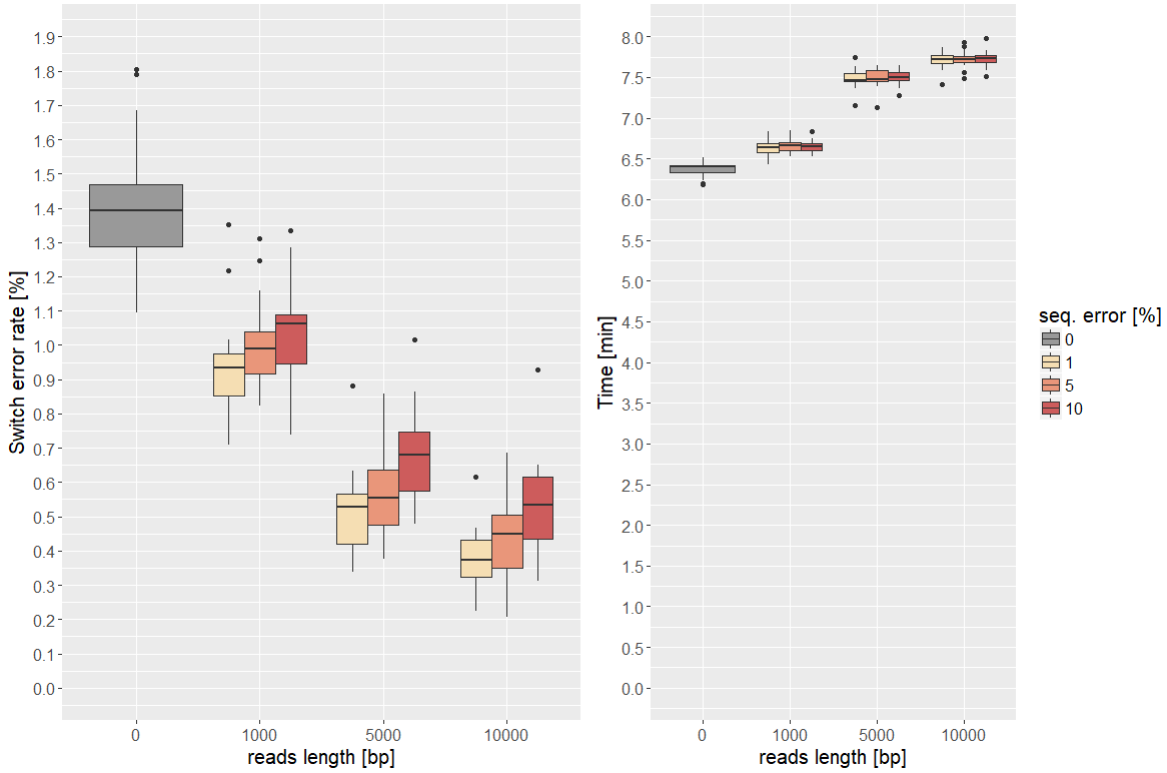


Figure 7.1: Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 15 and for $P = 100$.

7.1.3 Scenario 3: Low coverage

Next, we tested how well the algorithm performs on a read set containing long reads with a low coverage. In particular, we set the dataset coverage to 5 for reads of length $10k$. These values better correspond to standard datasets produced by third generation sequencing platforms. We used the same read panels as for the previous scenarios, only down-sampled to the desired coverage. We compared results obtained for sequencing error rates of 1%, 5% and 10% with results obtained for down-sampled dataset with the same parameters. The results of the experiment are illustrated in Fig. 7.3.

We can see from the left side plot in Fig. 7.3 that phasing performed on the dataset with coverage 5 results in a switch error rate higher on average than phasing on the dataset with coverage of 15. In particular, the algorithm achieved mean switch error of 0.37%, 0.43%, 0.53% for the dataset with coverage 15 and switch error of 0.43%, 0.52%, 0.67% for the dataset with coverage 5 for sequencing error of 1%, 5% and 10%, respectively and read length of $10k$ *bp*. Nevertheless, accuracy of the algorithm for the dataset with the low coverage was higher than accuracy of the algorithm when no read panel was supplied (the gray box on the left side plot), for all settings of the sequencing error rate. Importantly, this also holds for read set with high sequencing error rate of 10%, for which the algorithm still performs more than twice as better (1.4% vs 0.67%).

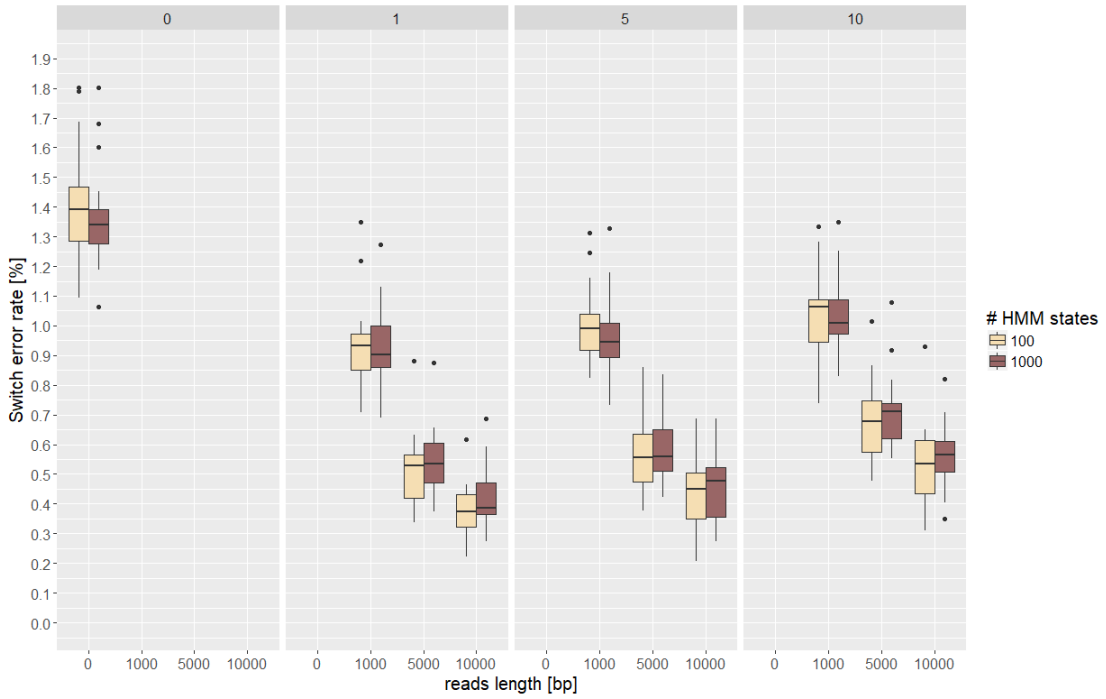


Figure 7.2: Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 15 and for $P = 100$ and $P = 1000$.

Although the performance of the algorithm had slightly decreased, the run time had dropped, as it might be seen from the right side plot in Fig. 7.3. For the the low coverage dataset, the run time of phasing without a read panel (the gray box of the right side plot) is nearly identical to run time of the algorithm with low-coverage read panel, regardless of the sequencing error rate. In particular, it took 6.36 minutes for phasing with no read panel and 6.9, 6.9, 6.95 minutes for phasing with a read panel, for sequencing error of 1%, 5% and 10%, respectively.

7.1.4 Scenario 4: High sequencing error rate

Additionally, we simulated one more dataset, containing long reads ($10k$ bp) with sequencing error rate of 20%. Achieved switch error rate for the dataset with coverage of 15 and 5 is illustrated in Fig. 7.3. We can see from the figure that accuracy is higher for phasing with a read set with 20% error rate than accuracy obtained achieved by the phasing without a read panel. However, we can see that the resulting switch error rate dropped in comparison with the result achieved for the lower error rates.

7.1.5 Read panel construction

CPU time of the hybrid phasing algorithm reported earlier in this section does not include time required to create a *.var* file. The reason is that the tool is independent and separate

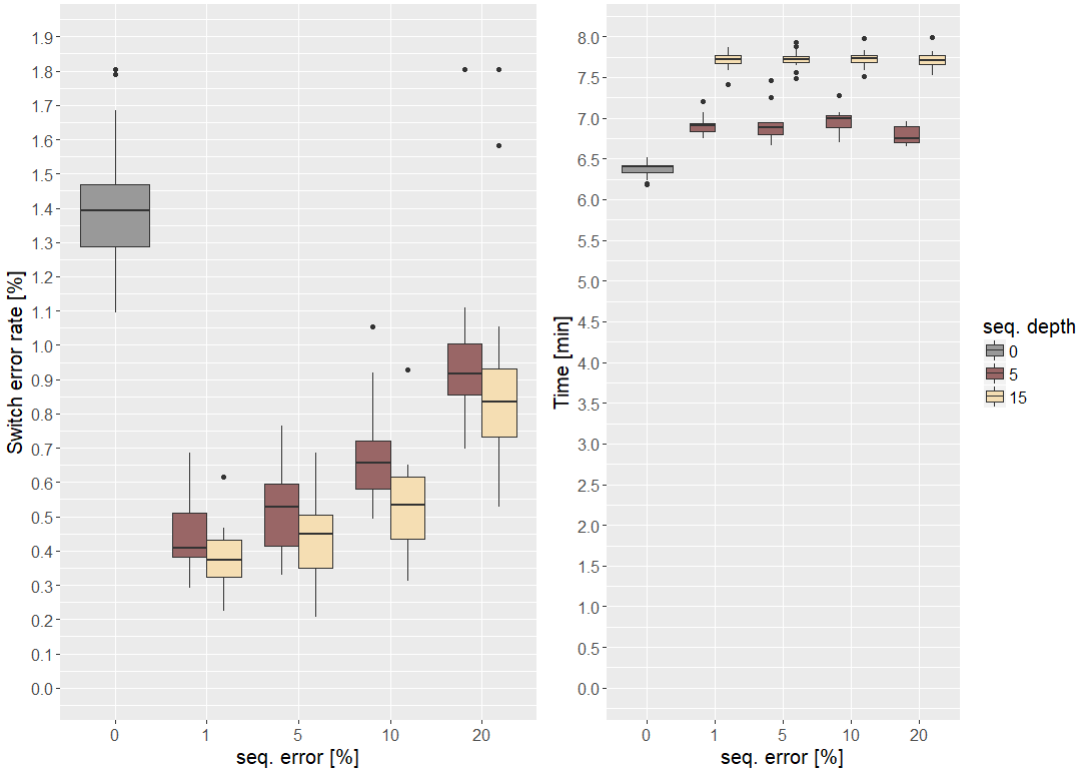


Figure 7.3: Box plots of the relation between the read set parameters (length and sequencing error rate) and the switch error rate and time, respectively, for the dataset with depth 5 and 15 and for $P = 100$.

from the main algorithm cycle. Moreover, construction of a read panel for a single sample takes less than a minute and is almost independent on the reads' parameters. The time is also influenced by choice of the programming language.

CPU time required for constructing datasets described in the first scenario with coverage of 15 and 5 are illustrated in Fig. 7.4. Note that the time was measured when 9 threads were running in parallel.

7.2 Discussion

The results presented in the previous section underline the ability of the hybrid algorithm to achieve lower switch error rate when a read panel is utilized. From the results of the experiments we conclude that a decrease in the switch error rate was achieved for all examined reads' parameters. Namely, for datasets of coverage 5 and 15 containing reads of length $1k$, $5k$ and $10k$ *bp* and sequencing error rates up to 20%.

We concluded that reads length is the primary factor increasing the algorithm accuracy. We observed that given a dataset with low sequences error rate, the algorithm was successful at regions where variant density was higher. Specifically, it was able to increase accuracy where distance between variants was less or equal to the reads length. We illustrate the

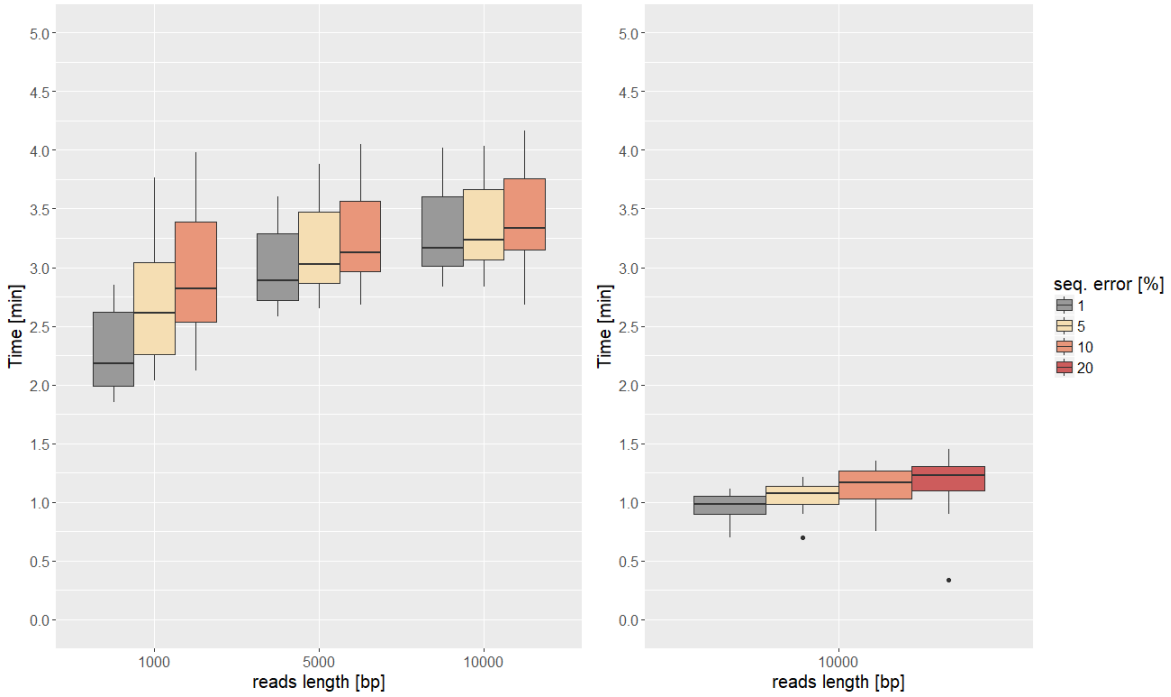


Figure 7.4: Box plots of the relation between the read set parameters (length and sequencing error rate) and *.var* creation time for the dataset with depth 15 and 5, respectively.

distance between start and end positions of the switch errors the algorithm made given reads of length $1k$, $5k$ and $10k$ for sample with id $HG00099$ in Fig. 7.5.

Surprisingly, errors the algorithm made when a read panel was supplied had a small intersection with the set of errors made by the algorithm when no read panel was given than we had expected. We observed a pattern where a group of shorter errors made by the phaser when no read panel was given was followed by a single or a smaller group of errors made when a read panel was supplied, independently on the reads' length. This underlines the different nature of errors the algorithm tends to make in those two settings.

We also noticed that the algorithm had a tendency to fail in variants which are INDELs. For instance, the proportion of misphased INDELs was 35% over all errors for dataset with sequencing error rate of 1% and 50% for dataset with sequencing error rate of 20% for read length of $10k$ for the sample $HG00096$. That might seem odd as INDELs constitute less then 10% of all variants in the human genome. We assume it might be caused by ambiguous alignment of reads near the variants.

We observe from the experiments that the algorithm does not gain a substantial increase in run time in comparing with run time it requires when no read panel is supplied. Moreover, gain in the CPU time constituted only around 6.5% in average of the total algorithm runtime when phasing with a read panel of coverage 5 was performed.

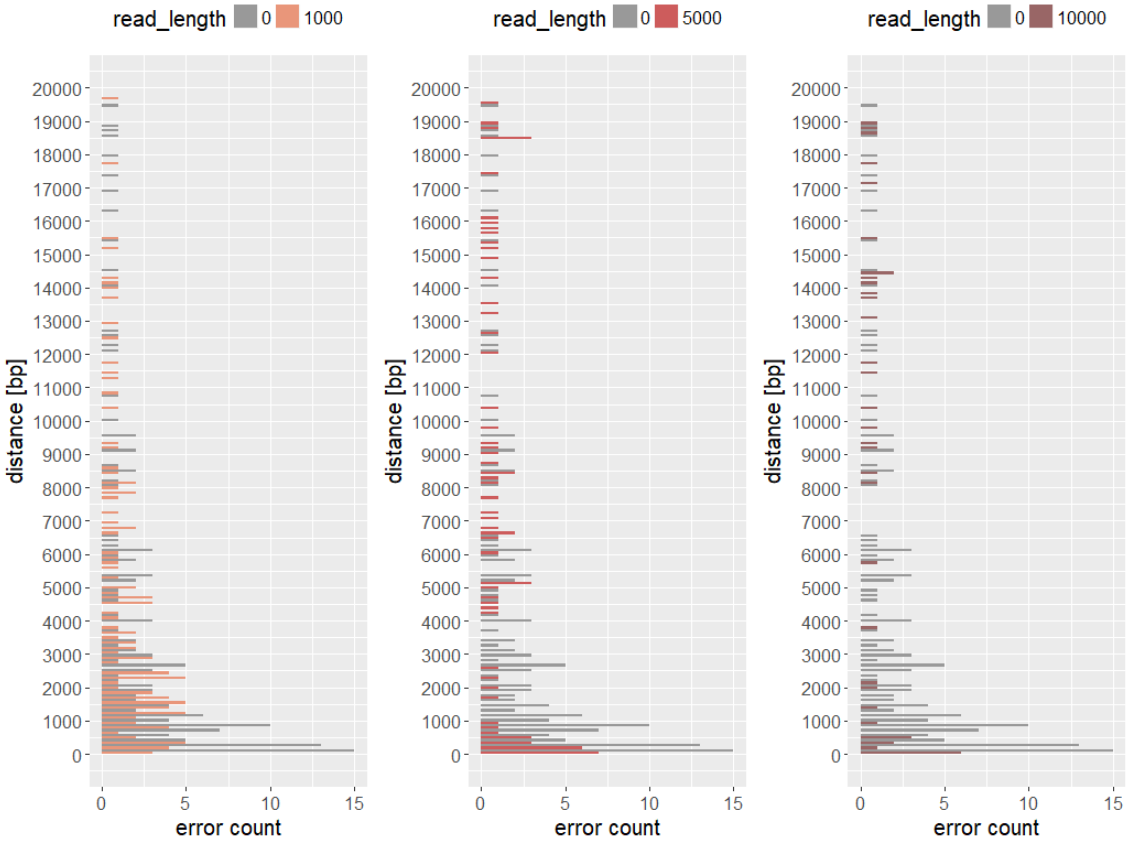


Figure 7.5: Histogram of distances between the start and end of switch error coordinates in the genome of sample $HG00099$ made by the phasing algorithm when no read panel was given and when a dataset with reads of length $1k$, $5k$ and $10k$ and 1% sequencing error rate was given.

We had also examined how number the of HMM states influences the algorithm performance and observed that there was no gain in accuracy for number of states $P = 100$ and $P = 1000$. On the other hand, run time of the algorithm depends linearly on P (as it was reported in Sec. 5.3). However, the result might be influenced by the value of the expected IBD length (not tested in this work).

7.3 Conclusion

To conclude, the hybrid phasing algorithm had demonstrated capability to gain higher phasing accuracy when a read panel is utilized in various settings. This makes the algorithm suitable for use in any pipeline where reads are available and a reference panel is present. We had demonstrated that the algorithm is highly accurate for relatively small reference panel which might be beneficent for instance for other species than humans for which there is no extensive reference panel.

The proposed read model is similar to SHAPEIT2 read model, except that our HMM

iterates between positions, while SHAPEIT2 HMM iterates over blocks of the fixed length. This makes us assume that the accuracy of the presented algorithm is comparable or even higher than the accuracy of the SHAPEIT2 algorithm.

Also, since the implemented algorithm shares main features with the Eagle2 algorithm, we expect the algorithms perform similarly in terms of both accuracy and CPU time when read panel is not given. The time complexity of the presented algorithm is only linearly dependent on the read set coverage and, as it was established, the algorithm remains extremely efficient in terms of CPU time when a read panel is supplied.

7.4 Future work

The proposed read model allowed to increase the algorithm accuracy by combining information about read alignment and read base qualities. We suppose that extension of the model by alignment qualities and prior expectations on sequencing error rate might increase the model accuracy especially when reads with high error rates are supplied.

We also plan to test the algorithm on real datasets produced by the second and third generation sequencing platforms and compare the results with other hybrid phasing algorithms. Additionally, we suggest examining the tendency of the algorithm to misphase variants which are INDELs and proposing methods to avoid the behavior.

Bibliography

[BBa] V. Bansal and V. Bafna, *Hapcut: an efficient and accurate algorithm for the haplotype assembly problem*, Bioinformatics.

[BBb] B.L. Browning and S.R. Browning, *Haplotype phasing: Existing methods and new developments*, Nat Rev Genet.

[BBc] ———, *Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering*, Am J Hum Genet.

[BBd] ———, *A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals*, Am J Hum Genet.

[BDK+] P. Bonizzoni, R. Dondi, G.W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria, *On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes.*, Comput Biology.

[Boa12] Editorial Board, *Concise dictionary of science*, V&s Publishers, 2012, p137.

[Chi] H. Chial, *Dna sequencing technologies key to the human genome project*, Nature Education.

[CMC+] S.E. Castel, P. Mohammadi, W.K. Chung, Y. Shen, and T. Lappalainen, *Rare variant phasing and haplotypic expression from rna sequencing with phaser*, Nature Communications.

[CML16] B. Cox, P.D. Moore, and R. Ladle, *Biogeography: An ecological and evolutionary approach*, Wiley-Blackwell; 9 edition, 2016.

[Cona] The 1000 Genomes Project Consortium, *A global reference for human genetic variation.*

[Conb] The Haplotype Reference Consortium, *A reference panel of 64,976 haplotypes for genotype imputation*, Nature Genetics.

[DAAea] P. Danecek, A. Auton, G. Abecasis, and et al., *The variant call format and vcftools*, Bioinformatics.

[DCZ] O. Delaneau, C. Coulonges, and J.F. Zagury, *Shape-it: new rapid and accurate algorithm for haplotype inference*, BMC bioinformatics.

[DHCea] O. Delaneau, B. Howie, A.J. Cox, and et al., *Haplotype estimation using sequencing reads*, The American Journal of Human Genetics.

[DHM+] J. Duitama, T. Huebsch, G. McEwen, E.K. Suk, and M.R. Hoehe, *Refhap: A reliable and fast algorithm for single individual haplotyping*, ACM.

[Dur] R. Durbin, *Efficient haplotype matching and storage using the positional burrows–wheeler transform (pbwt)*, Bioinformatics.

[FGC] W.G Feero, A.E. Guttmacher, and F.S. Collins, *Genomic medicine — an updated primer*, N Engl J Med.

[HC] J.M. Heather and B. Chain, *The sequence of sequencers: The history of sequencing dna*, Genomics.

[HFB] D.G. Hert, C.P. Fredlake, and A.E. Barron, *Advantages and limitations of next-generation sequencing technologies: a comparison of electrophoresis and non-electrophoresis methods*, Electrophoresis.

[HHE] D. He, B. Han, and E. Eskin, *Hap-seq: An optimal algorithm for haplotype phasing with imputation using sequencing data*, Journal of computitional biology.

[HS] R.R. Haraksingh and M.P. Snyder, *Impacts of variation in the human genome on gene regulation*, Journal of Molecular Biology.

[Ill] Inc. Illumina, *An introduction to next-generation sequencing technology.*

[Kea] A. Kong and et al., *Detection of sharing by descent, long-range phasing and haplotype imputation*, Nature Genetics.

[KGE] M. Kchouk, J.F. Gibrat, and M. Elloumi, *Generations of sequencing technologies: From first to next generation*, Biology and Medicine.

[Kul] V. Kuleshov, *Probabilistic single-individual haplotyping*, Bioinformatics.

[LCC] S. Lin, A. Chakravarti, and D. Cutler, *Haplotype and missing data inference in nuclear families*, Genome Res.

[LD] H. Li and R. Durbin, *Fast and accurate long-read alignment with burrows-wheeler transform.*, Bioinformatics.

[LDPea] P.R. Loh, P. Danecek, P.F. Palamara, and et al., *Reference-based phasing using the haplotype reference consortium panel*, Nat. Genet.

[LS] N. Li and M. Stephens, *Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data*, Genetics.

[MHM+] J. Marchini, B. Howie, S. Myers, G. McVean, and P. Donnelly, *A new multipoint method for genome-wide association studies by imputation of genotypes*, Nature Genetics.

[MMPD] J.M. Mullaney, R.E. Mills, W.S. Pittard, and S.E. Devine, *Small insertions and deletions (indels) in human genomes.*

[NLS] C.M. Nievergelt, O. Libiger, and N.J. Schork, *Generalized analysis of molecular variance*, PLoS Genet.

[PMea] M. Patterson, T. Marschall, and et al., *Whatshap: Weighted haplotype assembly for future-generation sequencing reads*, Journal of Computational Biology.

[PZD+] Y. Pirola, S. Zaccaria, R. Dondi, G.W. Klau, N. Pisanti, and P.s Bonizzoni, *Hapcol : accurate and memory-efficient assembly from long reads*, Bioinformatics.

[SHM+] Z.D. Stephens, M.E. Hudson, L.S. Mainzer, M. Taschuk, M.R. Weber, and R.K. Iyer, *Simulating next-generation sequencing datasets from empirical mutation and sequencing models*, PLoS ONE.

[SS] M. Stephens and P. Scheet, *A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase.*, Am J Hum Genet.

[SSD] M. Stephens, N.J. Smith, and P. Donnelly, *A new statistical method for haplotype reconstruction from population data*, Am J Hum Genet.

[SSEK] S. Song, E. Sliwerska, S. Emery, and J.M. Kidd, *Modeling human population separation history using physically phased genomes*, Genetics.

[Szk] M. Szklo, *Population-based cohort studies*, Epidemiologic Reviews.

[TBT+] R. Tewhey, V. Bansal, A Torkamani, E.J. Topol, and Schork N.J., *The importance of phase information for human genomics*, Nat Rev Genet.

[Tho] E.A. Thompson, *The ibd process along four chromosomes*, Theor Popul Biol.

[XWJ] M. Xie, J. Wang, and T. Jiang, *A fast and accurate algorithm for single individual haplotyping*, BMC Systems Biology.