Fast and accurate detection of Identical-By-Descent segment sharing in large genomic datasets



Juba Nait Saada
St Cross College
University of Oxford

A thesis submitted for the degree of
Master of Science by Research
Michaelmas 2019

It is my genuine gratefulness
and my warmest regard
that I dedicate this work
to Belkacem and Karima,
for their unlimited love and devotion ;
to Macicilia, Thiziri and Antinia,
for their endless support;
to Zaina, Ferroudja and Linda,
for their kindness and affection.

Acknowledgements

I would first like to thank my advisor, Pier Palamara, for his guidance, his continuous support, his motivation, his enthusiasm and for having introduced me to this incredible field. He consistently offered me advice and insisted in this work to be my own. His door was always open to me whenever I had difficulties in my work. Thank you for teaching me so much and for making my first insight into research such a nice and pleasant experience. I would also like to thank my thesis committee for the time dedicated to my work.

I am particularly grateful to my sisters and my parents for their constant support and faith in me. I would like to thank Anthony, without whom nothing would have been possible. My sincere thanks also goes to Victor for this incredible year in Oxford and to Oumayma for always being there for me. Many thanks to my friends Aymen, Clement, Kadem and Kais for their support. Finally, some special words of gratitude go to my friends Amale, Cecile, Eugenie, Justine and Maryam.

Abstract

Identical-By-Descent (IBD) segments are a fundamental measure of genetic relatedness and IBD detection is of great interest in many applications, including demographic inference and haplotype-based association. Algorithms that scale to large datasets (more than 10.000 diploid individuals), such as GERMLINE and RaPID sacrifice modeling accuracy in favor of computational speed. Efficient model-based algorithms for IBD detection such as RefinedIBD, on the other hand, enable detecting shorter IBD segments, but do not scale as well in large data sets. In this thesis, we present two new approaches, called PASMC and GASMC, aiming to bridge the gap between these two paradigms. We introduce the basics of coalescent theory to understand IBD segments detection and then provide an overview of a state-of-the art methods for detecting IBD segments, comparing the most commonly used and recent existing methods. We finally present PASMC and GASMC. Both methods utilize efficient string matching to detect candidate IBD segments and subsequently apply the recently developed ASMC algorithm based on coalescent HMM to filter out candidate segments that are unlikely to be IBD.

Contents

1 Introduction **1**

1.1 Basic definitions . . . 1
1.2 Identical By Descent . . . 2
1.3 Genetic relatedness in the modern era . . . 3
1.4 Coalescent theory: The Wright-Fisher model . . . 4
1.5 Discrete-time coalescent . . . 5
1.6 Continuous-time coalescent . . . 6
1.7 The Wright-Fisher model with recombination . . . 7
1.7.1 Definition . . . 7
1.7.2 Adding recombination to the model . . . 8
1.8 Length of an IBD segment . . . 9
1.9 IBD segments simulator . . . 10
1.10 The Wright-Fisher model with mutation . . . 10
1.11 IBD segments simulator with mutation . . . 13
1.12 Simulation . . . 13

2 State of the art **16**

2.1 Overview of genomic data . . . 16
2.2 String matching methods . . . 17
2.2.1 Definition . . . 17
2.2.2 GERMLINE . . . 18
2.2.3 RaPID . . . 19
2.3 Model-based methods . . . 20
2.3.1 Overview . . . 20
2.3.1.1 Hidden Markov Model . . . 20
2.3.1.2 Forward-Backward Algorithm . . . 21
2.3.1.3 Viterbi Algorithm . . . 23
2.3.1.4 IBD score . . . 23

2.3.2 RefinedIBD . . . 23
2.3.3 ASMC . . . 25
2.4 Comparison . . . 26
2.4.1 Simulation of data . . . 26
2.4.2 Accuracy . . . 27
2.4.2.1 Precision-Recall curve vs ROC curve . . . 27
2.4.2.2 Thresholds . . . 28
2.4.2.3 IBD definition . . . 28
2.4.2.4 Grid search . . . 29
2.4.2.5 Results . . . 29
2.4.3 Time complexity . . . 34
2.4.4 The accuracy-scalability trade-off . . . 34

3 Preprocessed Ascertained Sequentially Markovian Coalescent 36
3.1 Method . . . 36
3.1.1 Identification of IBD candidates . . . 36
3.1.1.1 Positional Burrows-Wheeler transform (PBWT) . . . 36
3.1.1.2 Multiple runs with random projections . . . 43
3.1.2 Validation of IBD candidates . . . 44
3.1.3 Computational speed-ups . . . 44
3.1.4 Parameters . . . 47
3.2 Benchmarking . . . 48
3.2.1 Accuracy . . . 48
3.2.2 Time complexity . . . 51
3.2.3 Memory . . . 51
3.3 Discussion . . . 52
3.3.1 Downsides of PASMC . . . 52
3.3.2 Future directions: a new and more promising approach . . . 52

4 Germline - Ascertained Sequentially Markovian Coalescent 54
4.1 Method . . . 54
4.2 Benchmarking . . . 55
4.3 Discussion . . . 60

5 Conclusions 63

A IBD segments simulator - source code 64

B GASMC Manual **70**
B.1 Installation 70
B.2 Run and input 70
B.3 Output 71
B.4 Options 71

Bibliography **72**

List of Figures

1.1 A pedigree structure illustrating the process of Identical By Descent segment. 3
1.2 Discrete-time coalescent for a population of 5 genes (figure adapted from [12]). 6
1.3 Coalescent tree for a population of 5 genes with the discrete (left) and corresponding continuous time (right) (figure adapted from [12]). . . . 7
1.4 The Holliday Junction between two homologous chromosome during prophase I (figure adapted from [38]). 7
1.5 Heteroduplex regions. DNA strands is swapped between the two homologous chromosomes (figure adapted from [38]). 8
1.6 Gene conversion event (left), recombination event (right) (figure adapted from [38]). 8
1.7 Length of two individuals' IBD segments in function of the time their MRCA simulated with Algorithm 2. 14
1.8 Statistics from the simulation based on Algorithm 2. 14
1.9 Coalescent and recombination times' distributions from the simulation. 15
1.10 Frequency of mutations per IBD segment in the simulation. 15

2.1 Hidden Markov Model where X_i is the hidden state and O_i is the observation at time step $t = i$. In the case of IBD segments detection, each time step represents a chromosomal position. 22
2.2 Building observation process in ASMC for each pair of haplotypes. . 26
2.3 Accuracy results obtained for IBD detection within the past 25 generations on 10 independent datasets. 30
2.4 Accuracy results obtained for IBD detection within the past 50 generations on 10 independent datasets. 31
2.5 Accuracy results obtained for IBD detection within the past 100 generations on 10 independent datasets. 31

2.6 Accuracy results obtained for IBD detection within the past 250 generations on 10 independent datasets. 32
2.7 Accuracy results obtained for IBD detection within the past 400 generations on 10 independent datasets. 32
2.8 Accuracy results obtained for IBD detection within the past 600 generations on 10 independent datasets. 33
2.9 Time complexity for every IBD time threshold considered. 35

3.1 Building observations in ASMC vs PASMC for three different haplotypes. x represents a SNP not identified as IBD by PASMC. In this example, the minimum length is 5 SNPs, the window size is 1, the number of runs is 1 and the minimum number of successes is 1. ASMC considers all $\binom{3}{2}$ pairs of individuals and the whole chromosomal segment for each of them, while PASMC builds observations only on IBD candidate segments. 45
3.2 Accuracy results obtained with an IBD time threshold of 100 generations on 10 independent datasets. The average common recall range is [0.01, 0.50]. 49
3.3 Accuracy results obtained with an IBD time threshold of 250 generations on 10 independent datasets. The average common recall range is [0.00, 0.26]. 49
3.4 Accuracy results obtained with an IBD time threshold of 400 generations on 10 independent datasets. The average common recall range is [0.00, 0.16]. 49
3.5 Time complexity for every IBD time threshold considered. 50
3.6 Memory complexity of PASMC algorithm. 52

4.1 Accuracy results obtained with an IBD time threshold of 25 generations on 10 independent datasets. The average common recall range is [0.07, 0.82]. 57
4.2 Accuracy results obtained with an IBD time threshold of 50 generations on 10 independent datasets. The average common recall range is [0.03, 0.71]. 57
4.3 Accuracy results obtained with an IBD time threshold of 100 generations on 10 independent datasets. The average common recall range is [0.01, 0.53]. 57

4.4 Accuracy results obtained with an IBD time threshold of 250 generations on 10 independent datasets. The average common recall range is [0.00, 0.29]. 58
4.5 Accuracy results obtained with an IBD time threshold of 400 generations on 10 independent datasets. The average common recall range is [0.00, 0.25]. 58
4.6 Accuracy results obtained with an IBD time threshold of 600 generations on 10 independent datasets. The average common recall range is [0.00, 0.21]. 58
4.7 Precision-recall curves randomly sampled from 10 independent simulated datasets. 59
4.8 Time complexity for every IBD time threshold considered. 62
4.9 Memory complexity of GASMC algorithm. 62

Abbreviations

ASMC	Ascertained Sequentially Markovian Coalescent
CSFS	Conditioned Sample Frequency Spectrum
GASMC	Germline Ascertained Sequentially Markovian Coalescent
HMM	Hidden Markov Model
IBD	Identical By Descent
IBS	Identical By State
LOD	Logarithm Of Odds
MAP	Maximum A Posteriori
MRCA	Most Recent Common Ancestor
PBWT	Positional Burrows-Wheeler Transform
PASMC	Preprocessed Ascertained Sequentially Markovian Coalescent
SNP	Single-Nucleotide Polymorphism

Chapter 1

Introduction

This chapter only gives some insight into coalescent theory to help the reader understand better the concepts related to IBD detection. I strongly recommend the reader to read the "Gene Genealogies, Variation and Evolution" book [12] on which this chapter is based. We will go through some definitions and basics of gene genealogies, introduce new concepts and finally simulate IBD segments among two individuals.

1.1 Basic definitions

The genetic instructions used in the functioning and reproduction of all known living organisms and many viruses are contained in a molecule called deoxyribonucleic acid, or DNA, first isolated by Friedrich Miescher in 1869 [7]. DNA is composed of two strands, known as polynucleotides, each of them composed of simpler monomeric units called nucleotides. Its molecular structure was identified by James Watson and Francis Crick in 1953 [37]. Each nucleotide is composed of one of four nitrogen-containing nucleobases (cytosine [C], guanine [G], adenine [A] or thymine [T]), a sugar called deoxyribose, and a phosphate group. Each base couples with its complement (A with T and C with G) which forms a base pair. The order of succession of the nucleobases in each strand forms the sequence of the strand. A specific genomic location may be referred to as a site or a locus. We define a gene as set of sites encoding a protein. Due to the occurrence of mutations and other rare genomic events, different versions of a gene may exist in a population, referred to as alleles. We call single-nucleotide polymorphism (abbreviated to SNP) a mutation reaching high frequency at a specific location and is informative to discriminate genomic variation across individual or underlies differences in our susceptibility to disease. Within eukaryotic cells, DNA is organized into long structures called chromosomes. Humans have 23 pairs of chromosomes (22 pairs of autosomes and one pair of sex chromosomes),

giving a total of 46 per cell. Because two copies of each chromosome is stored in cells, humans are diploid organisms. Cells containing only one copy of each chromosome refer to haploid organisms. During the process of meiosis discovered by the biologist Oscar Hertwig in 1876 [1], diploid individuals produce gametes (egg cells for female individuals and sperm cells for male individuals) for sexual reproduction. The meiosis process is a type of cell division that reduces the chromosome number by half. The gametes contain only one copy of each chromosome formed by mixing the two original existing copies [27], which is called a recombination event. Two gametes, one from the maternal parent and one from the paternal parent, fuse during fertilization, creating a diploid cell with a complete set of paired chromosomes.

1.2 Identical By Descent

Two DNA segments are Identical By State (IBS) if they have identical nucleotides at every location. They are said to be Identical By Descent (IBD) if one of them is descended from the other or if both are descended from a third ancestral segment. "Descent" refers to DNA replication and transmission from parent to offspring, while "identical" refers to the genetic relatedness between the two individuals, although a small number of mutations and other rare genomic events may occur on each segment during the transmission process. As detailed later on in this chapter, the length of an IBD segment directly depends on the distance in generations to the common ancestor: as we will see through the equations, long segments are generally associated to recent common ancestor while short co-inherited segments are from a remote ancestor. However, we will also see that, under some assumptions, the number of mutations per IBD segment doesn't depend on how long ago the common ancestor has lived (see Equation 1.8). Then, every IBD segment will have on average the same number of mutations. Consequently, very recent IBD segments will be more identical to the original ancestor's segment than segments co-inherited from a very remote ancestor because both will have on average the same number of mutations but the former will be much longer. IBD denotes pairwise genetic relatedness but can easily be extended to a group of individuals. There are three common ways to define IBD: a pedigree definition, a coalescent definition and an ancestral-segment definition. They respectively refer to "individuals related by a known family structure or pedigree; alleles descended from a common ancestral allele within a specified time ... and genomic tracts of strong genetic similarity demarcated by recombination events" [36]. We illustrate the predigree definition of IBD segment in Figure 1.1. In

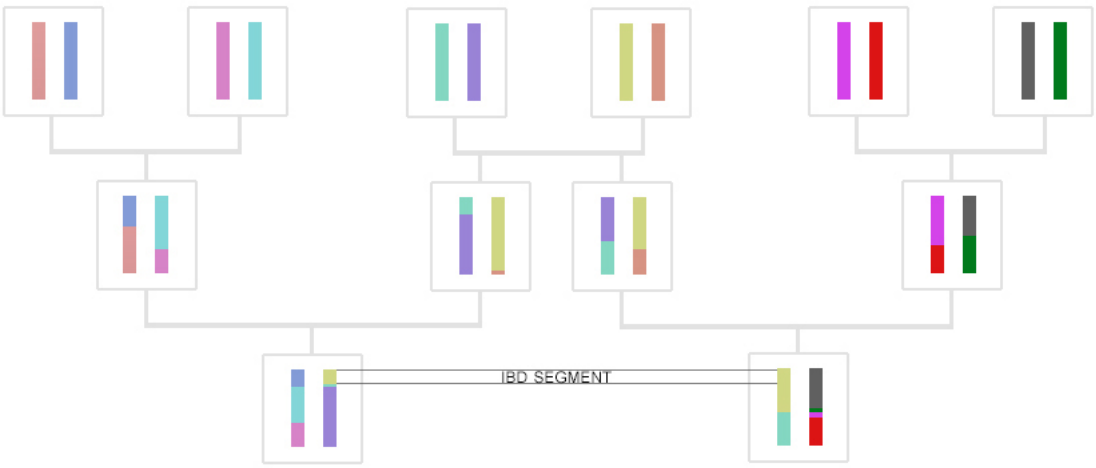


Figure 1.1: A pedigree structure illustrating the process of Identical By Descent segment.

this example, a pair of cousins share two common ancestors that lived two generations ago. Only one pair of chromosome and one strand of each chromosome are illustrated here. At each generation, offspring inherit a chromosome copy of one of the two chromosomes from each parent. However, such inherited copies result from the meiotic events that generate germ cells, during which recombination events may mix the original chromosome copies. This results in a new chromosome with DNA segments from both chromosomes of the original pair. Each offspring then carries genetic information from all four parental chromosomes. After two generations, the pair of cousins inherits DNA segments from their ancestors and one of them overlaps in a region (colored in yellow), which forms an IBD segment [27]. In this thesis, we will focus on the coalescent definition of IBD, which is the one used for our method PASMC presented in Chapter 3.

1.3 Genetic relatedness in the modern era

In 1990, the National Institute of Health and the Department of Energy in the US decided to launch the Human Genome Project with the goal of determining the sequence of the 3 billion nucleotide base pairs of the whole human DNA. More than 350 laboratories in 18 different countries were involved in the project. However in 1998, the Celera Genomics company was built in association with Perkin-Elmer, a multinational leader specialized in DNA sequencing equipment which allowed them to perform sequencing in a much more efficient way (with a powerful capacity 50

times higher than the existing laboratories) [8]. In 2001, both Celera Genomics and the public consortium published their results (see [35] and [5]). In 2003, the completion of the human genome sequencing was announced which resulted in an important development of DNA sequencing technologies with massive parallel platforms. The average cost for obtaining a full human genome DNA sequence dropped exponentially. Furthermore, the advent of the "next-generation" sequencing technologies in 2008 replaced Sanger-based sequencing technologies and decreased the cost of DNA sequencing [27]. From 2001 to 2012, the price of sequencing a single human genome dropped from 100 million dollars to a few thousand dollars [24]. Developments in sequencing technology made access to genotype data easier and cheaper. Consequently, genotype data of various populations are more and more available, generated both from genome-wide SNP array or whole genome sequencing. Different projects are currently deploying large datasets in order to make them accessible to the research community. The 1000 Genomes Project [6] created a very large public catalogue of human variation and genotype data of more than 2500 individuals from 2008 to 2015. The UK Biobank [4] recently released genotype data of around 500 000 individuals. Available genotype data keeps increasing and will probably reach hundreds of thousands up to millions in the future. Because data is becoming more available, measuring genetic relatedness between individuals is now possible and IBD detection is of great interest in many applications. Every group of human beings in the world are connected through very complex genealogical relationships. Some of them have at a specific genomic site a common ancestor who lived a few generations ago, while others share a common ancestor who lived thousands of generations ago. Many individuals are not aware of their genetic relatedness and detecting IBD segments could help people understand their origins and identify unknown familial ties and distant cousins. But accurate detection of these IBD regions is interesting in many other genomic analyses: IBD segments can reveal the presence of disease-causing mutations [3] and fine-scale demographic history [26].

1.4 Coalescent theory: The Wright-Fisher model

In order to find genetic relatedness, our approach in this work will be based on coalescent theory. Coalescent theory describes how gene variants sampled from a population have originated from a common ancestor. In the case of the IBD segments detection, we are interested in the most recent common ancestors (MRCA) between individuals in every chromosomal region. There are many existing mathematical

models to describe the genealogical relationship among genes. We will focus on the one introduced by Wright [40] and Fisher [10]. This model makes many assumptions to make it less mathematically complex. Among them:

- Discrete and non-overlapping generations;
- Haploid individuals;
- Constant population size;
- No social or geographical structure;
- No natural selection.

1.5 Discrete-time coalescent

Let us assume a population of $2N$ individuals. Let T_k denote the time in generations before the MRCA between two genes is found among k genes. The probability that k genes have k different ancestors in the previous generation is given by:

$$\begin{aligned}\frac{2N}{2N} \times \frac{2N-1}{2N} \times \frac{2N-2}{2N} \times \cdots \times \frac{2N-k+1}{2N} &= \prod_{i=1}^{k-1} \left(1 - \frac{i}{2N}\right) \\ &= 1 - \sum_{j=1}^{k-1} \frac{j}{2N} + \mathcal{O}\left(\frac{1}{N^2}\right) \\ &= 1 - \binom{k}{2}\frac{1}{2N} + \mathcal{O}\left(\frac{1}{N^2}\right)\end{aligned}$$

Thus, we approximate:

$$P(T_k = 1) = \binom{k}{2}\frac{1}{2N} \tag{1.1}$$

Consequently,

$$\forall j \in \mathbb{N}^*, \quad P(T_k = j) = \binom{k}{2}\frac{1}{2N} \times \left(1 - \binom{k}{2}\frac{1}{2N}\right)^{j-1} \tag{1.2}$$

T_k has a geometric distribution with parameter $\binom{k}{2}\frac{1}{2N}$. An example is shown in Figure 1.1.

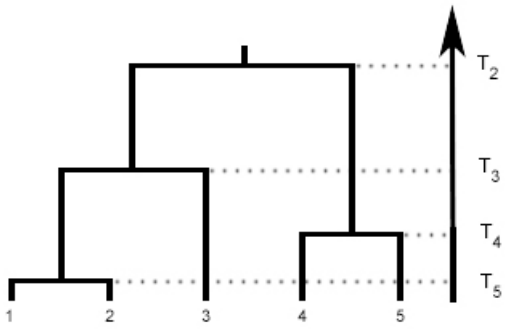


Figure 1.2: Discrete-time coalescent for a population of 5 genes (figure adapted from [12]).

1.6 Continuous-time coalescent

In order to make equations simpler, most mathematical models scale time such as one unit of time corresponds to the average time for two genes to find a common ancestor, which is $\mathbb{E}(T_2) = 2N$ (from Equation 1.2). Thus, we introduce t the continuous time:

$$t = \frac{j}{2N} \tag{1.3}$$

for j in $\mathbb{N}$ the discrete time in generations. Let T_k^c denote the continuous time before the MRCA between two genes is found among k genes.

$$\begin{aligned} P(T_k^c \leq t) &= P(T_k \leq j) \\ &= 1 - \left(1 - \binom{k}{2}\frac{1}{2N}\right)^j \\ &= 1 - \left(1 - \binom{k}{2}\frac{1}{2N}\right)^{2Nt} \\ &= 1 - \exp\left(2Nt\ln(1 - \binom{k}{2}\frac{1}{2N})\right) \\ &\simeq 1 - \exp\left(-\binom{k}{2}t\right) \qquad for \quad N \to \infty \end{aligned}$$

T_k^c has an exponential distribution with parameter $\binom{k}{2}$. In Figure 1.2, we present a coalescent tree with the discrete and continuous time.

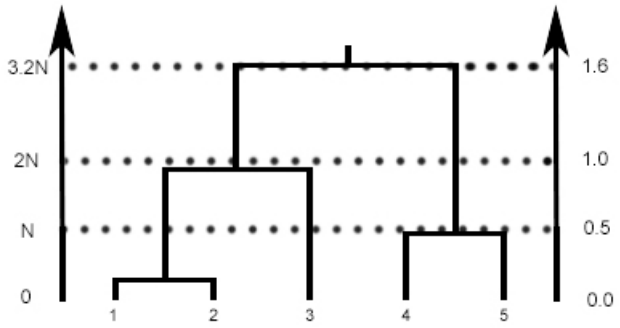


Figure 1.3: Coalescent tree for a population of 5 genes with the discrete (left) and corresponding continuous time (right) (figure adapted from [12]).

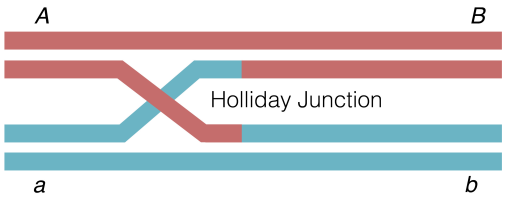


Figure 1.4: The Holliday Junction between two homologous chromosome during prophase I (figure adapted from [38]).

1.7 The Wright-Fisher model with recombination

1.7.1 Definition

During meiosis, and more specifically during prophase I, homologous chromosomes pair and exchange DNA. The first molecular model of the genetic exchange between homologous chromosomes was proposed by Robin Holliday in 1964 [18]. The Holliday model involves different steps. The two chromosomes are nicked at identical locations. Then, the strands from one side of the nick invade the homologuous chromosome. The invading strand is then linked to the original strand, forming a Holliday junction (Figure 1.4). The Holliday junction migrates from the original nick site and DNA strands are swapped between the two chromosomes, which creates heteroduplex regions (Figure 1.5). There are two different repair mechanisms. One of them results in a gene conversion event without crossing over, and the other one in a gene conversion event with crossing over. The former is commonly called a gene conversion, and the last one a recombination event (Figure 1.6). Gene conversion will be ignored in our model.

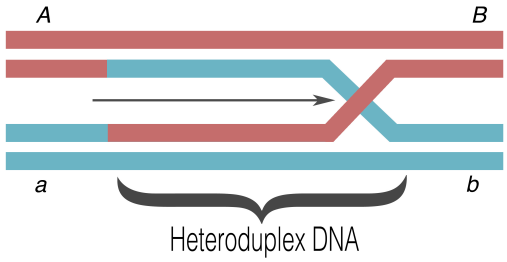


Figure 1.5: Heteroduplex regions. DNA strands is swapped between the two homologous chromosomes (figure adapted from [38]).

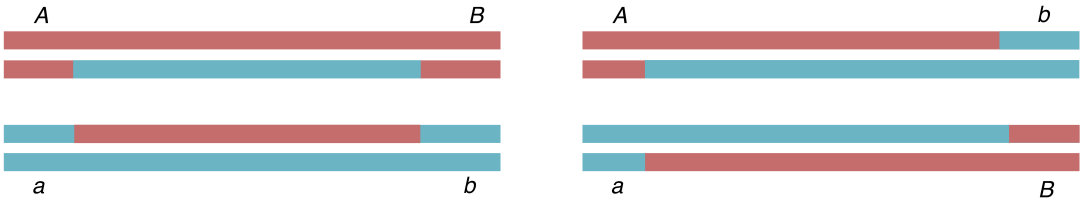


Figure 1.6: Gene conversion event (left), recombination event (right) (figure adapted from [38]).

1.7.2 Adding recombination to the model

Let us suppose a recombination event takes place in a specific chromosomal region with probability r, called the recombination rate. If it does, then the location where homologous chromosomes merged is chosen randomly along the paternal and maternal sequences. Based on what we did before, we can define the scaled recombination rate $\rho = 4Nr$. Let T_R denote the number of generations before the first recombination event occurs for two individuals, and T_R^c the corresponding continuous time.

$$P(T_R = j) = r(1-r)^{j-1} \tag{1.4}$$

T_R has a geometric distribution with parameter r. Once again, we derive the exponential distribution in continuous time from the geometric distribution in discrete time.

$$\begin{aligned} P(T_R^c \leq t) &= P(T_R \leq j) \\ &\simeq 1 - \exp\left(-\rho\frac{t}{2}\right) \qquad for \quad N \to \infty \end{aligned}$$

Consequently, T_R^c has an exponential distribution with parameter $\frac{\rho}{2}$.

1.8 Length of an IBD segment

Understanding the recombination event is crucial for IBD segments detection. Considering two individuals, if a common ancestor has been found at a coalescent time T_{col} on a specific site, we need to know how far can we go on the chromosome to still consider the segment as an IBD and then figure out the total length of the IBD segment. As long as a recombination event didn't occur between present time and T_{col} at the next location, we would still consider this next site as part of the IBD segment. Indeed, if a recombination event occurred at a time between now and T_{col}, the IBD segment would break because a different common ancestor has been found and the next sites have been inherited from this new common ancestor. This is referring to the sequentially Markovian coalescent' approximation (see [20] and [27]) and this is what we will assume in this thesis. Let L denote the discrete physical length (number of SNPs) of a chromosomal segment of two individuals sharing their most recent common ancestor at time T^c_{col}, where T^c_{col} is a scaled version of T_{col} representing the continuous time. For each individual, each site is subject to recombination event between now and T^c_{col} with probability $1 - \exp\left(-\rho\frac{T^c_{col}}{2}\right) \simeq T^c_{col} \times \frac{\rho}{2}$, where ρ is the scaled recombination rate. Thus, for each pair of individuals, each site is subject to recombination event with approximate probability $T^c_{col}\rho$. Then,

$$P(L = l|T^c_{col}) = T^c_{col}\rho(1 - T^c_{col}\rho)^{l-1} \tag{1.5}$$

L has a geometric distribution with parameter $T^c_{col}\rho$. We introduce $L_{cM} = L \times \rho$ the length in centiMorgan (cM) of the IBD segment, with the MRCA at T^c_{col}. Let us assume ρ is fixed. As before, we derive the exponential distribution in continuous time from the geometric distribution.

$$\begin{aligned} P(L_{cM} \leq l_{cM}|T^c_{col}) &= P(L \leq l|T^c_{col}) \\ &\simeq 1 - \exp\left(-T^c_{col}\rho l\right) \qquad for \quad N \to \infty \quad and \quad \rho \quad fixed \\ &\simeq 1 - \exp\left(-T^c_{col} l_{cM}\right) \end{aligned}$$

Consequently, L_{cM} has an exponential distribution with parameter T^c_{col}. The expected length of an IBD segment with a common ancestor who has lived T^c_{col} continuous generations ago is $\frac{1}{T^c_{col}}$ cM which means that short IBD segments are more likely to be associated to very ancient common ancestors while long segments are more likely to be inherited from recent ancestors.

1.9 IBD segments simulator

We now know the basic requirements to build our own coalescent simulator. Let us assume a constant population size N, a constant recombination rate ρ and a constant mutation rate μ. We denote the total length of the chromosome in cM by L_{tot}. We only consider two individuals here. We first do not consider mutations to make the model simpler (Algorithm 1).

Algorithm 1 IBD segments simulator without mutation

```
position ← 0
recombTime ← 0
coalTime ← 0
results ← new List[]
while position < L_tot do
    coalTime ← recombTime + sample from exponential distribution with parameter 1
    length ← sample from exponential distribution with parameter coalTime
    position ← position + length
    recombTime ← sample uniformly in [0,coalTime]
    results.append([position, coalTime, recombTime])
return results
```

The output of Algorithm 1 is a list of IBD segments. Each IBD segment is a list of three elements: the first element is the ending position of the IBD segment, the second one is the time of the MRCA and the third one is the time of the recombination event breaking the IBD segment. Note that this algorithm assumes the Sequentially Markovian Coalescent (SMC) model [20].

1.10 The Wright-Fisher model with mutation

In 1969, Motoo Kimura proposed the infinite sites model [15], which allows us to incorporate mutations in the coalescent model without too complex equations. This model relies on two main assumptions: the number of sites where mutations can occur is infinite and every new mutation occurs at a novel site. This model prevents every site to be affected than more than one mutation and this is justified by the fact that the total number of mutated sites in human DNA is extremely small compared to the total number of sites (all humans are about 99.4% identitcal [6]). To model mutation, each gene that is passed on between two generations is subject to a mutation event with probability μ, called the mutation rate. Let T_M denote the time in generations before a lineage experiences its first mutation.

$$P(T_M = j) = \mu(1-\mu)^{j-1} \tag{1.6}$$

T_M has a geometric distribution with parameter μ. Similarly, we denote the continuous time before a lineage experiences its first mutation by T_M^c. As before, we derive the exponential distribution from the geometric distribution.

$$\begin{aligned} P(T_M^c \leq t) &= P(T_M \leq j) \\ &\simeq 1 - \exp\left(-\theta\frac{t}{2}\right) \qquad where \quad \theta = 4N\mu \qquad for \quad N \to \infty \end{aligned}$$

T_M^c has an exponential distribution with parameter $\frac{\theta}{2}$, where θ is called the scaled mutation rate. A mutation can happen in any lineage and doesn't affect other lineages. Lineages are then independent. Each of them has an exponential distribution, so we can conclude the time for the first mutation to happen in n lineages $T_{M,n}^c$. $T_{M,n}^c$ has an exponential distribution with parameter $n \times \frac{\theta}{2}$. We can now focus on the number of mutations on an IBD segment which has been studied in [27]. Let M denote the number of mutations on an IBD segment of l sites, with a common ancestor that lived j generations in the past (in discrete time), a constant scaled recombination rate ρ and a constant population size. We can convert the length l in centiMorgans using $l_{cM} = l \times \rho$. Remember that μ denotes the probability that a site is subject to a mutation event when it is passed on from parent to offspring. Based on the previous results, the probability that a single site mutates from j generations (or t in the continuous time) to present time for both lineages is given by $T_{M,2}^c$, which is $1 - \exp(-\theta t) = 1 - \exp(-2j\mu)$. In practice, μ is in the order of 10^{-8} [32], so we can approximate $1 - \exp(-2j\mu) \simeq 2j\mu$. Then,

$$P(M = m | T_{col} = j, L = l) = \binom{l}{m}(2j\mu)^m(1-2j\mu)^{l-m} \tag{1.7}$$

M has a binomial distribution with parameters $(l, 2j\mu)$. Chromosomal segments contain a very large number of sites: human chromosomes range in size from about 50,000,000 to 300,000,000 base pairs [25]. Thus, we would have in practice $l >> 1$ and $2j\mu << 1$ (because $\mu \simeq 10^{-8}$). Let us define λ such as $\lambda = l \times 2j\mu$. We can now derive the continuous distribution of M from the binomial distribution when $l \to \infty$

and λ fixed (so $2t\mu \to 0$).

$$\begin{aligned}
\lim_{l\to\infty} P(M = m|T_{col} = j, L = l) &= \lim_{l\to\infty} \binom{l}{m} \left(\frac{\lambda}{l}\right)^m \left(1 - \frac{\lambda}{l}\right)^{l-m} \\
&= \lim_{l\to\infty} \frac{l!}{m!(l-m)!} \left(\frac{\lambda}{l}\right)^m \left(1 - \frac{\lambda}{l}\right)^{l-m} \\
&= \left(\frac{\lambda^m}{m!}\right) \lim_{l\to\infty} \frac{l!}{(l-m)!} \left(\frac{1}{l^m}\right) \left(1 - \frac{\lambda}{l}\right)^{l} \left(1 - \frac{\lambda}{l}\right)^{-m}
\end{aligned}$$

We can now take the limit of the right-hand side one term at a time.

$$\begin{aligned}
\lim_{l\to\infty} \frac{l!}{(l-m)!} \left(\frac{1}{l^m}\right) &= \lim_{l\to\infty} \frac{l(l-1)\dots(l-m)(l-m-1)\dots 1}{(l-m)(l-m-1)\dots 1} \left(\frac{1}{l^m}\right) \\
&= \lim_{l\to\infty} \frac{l(l-1)(l-2)\dots(l-m+1)}{l^m} \\
&= \lim_{l\to\infty} \frac{l}{l} \times \frac{l-1}{l} \times \frac{l-2}{l} \cdots \times \frac{l-m+1}{l} \\
&= 1 \\
\lim_{l\to\infty} \left(1 - \frac{\lambda}{l}\right)^{l} &= \lim_{l\to\infty} \exp\left(l \times \ln\left(1 - \frac{\lambda}{l}\right)\right) \\
&= \exp(\lambda) \\
\lim_{l\to\infty} \left(1 - \frac{\lambda}{l}\right)^{-m} &= 1
\end{aligned}$$

Finally, by taking the product of all limits,

$$\lim_{l\to\infty} P(M = m|T_{col} = j, L = l) = \left(\frac{\lambda^m \exp(\lambda)}{m!}\right)$$

M has a Poisson distribution with parameter $\lambda = l \cdot 2j\mu = \frac{l_{cM}}{\rho} \cdot 2j\mu = \frac{l_{cM}}{4Nr} \cdot 4Nt\mu$, where t is the continuous scaled version of j and r is the recombination rate. Consequently, the expected number of mutations for an IBD segment transmitted j generations ago in the continuous time is:

$$\mathbb{E}(M|T^c_{col} = t) = \frac{\mathbb{E}(L_{cM}|T^c_{col} = t)}{r} \times t\mu = \frac{1}{r \times t} \times t\mu = \frac{\mu}{r} \tag{1.8}$$

Note that the expectation of the IBD segment's length results from the previous paragraph. We can see here that the expected number of mutations in an IBD segment doesn't depend on how long ago the common ancestor has lived which might be a surprising result.

1.11 IBD segments simulator with mutation

We may now add mutations in the simulator (see Algorithm 2). As the previous algorithm, the output of Algorithm 2 is a list of IBD segments. Each IBD segment is a list of four elements: the first three elements are the three elements from Algorithm 1, and the last one is a list of sites' locations where a mutation occurred in one of the two lineages since the common ancestor passed on the nucleotide to the two considered individuals.

Algorithm 2 IBD segments simulator

position $\leftarrow 0$
recombTime $\leftarrow$ *0*
coalTime $\leftarrow$ *0*
results $\leftarrow$ *new List[]*
while position $< L_{tot}$ **do**
    coalTime $\leftarrow$ *recombTime + sample from exponential distribution with parameter 1*
    length $\leftarrow$ *sample from exponential distribution with parameter coalTime*
    position $\leftarrow$ *position + length*
    recombTime $\leftarrow$ *sample uniformly in [0,coalTime]*
    nbMutations $\leftarrow$ *sample from poisson distribution with parameter* μ/r
    mutations $\leftarrow$ *new List[]*
    for nbMutations **do**
        site $\leftarrow$ *sample uniformly in [position, length]*
        mutations.append(*site*)
    results.append([position, coalTime, recombTime, mutations])
return results

1.12 Simulation

We implemented Algorithm 2 in Python (Appendix A). The population size was set to 10 000 diploid individuals (20 000 haploid individuals), the total length of the chromosome was set to 50 cM, a constant mutation rate of 10^{-8} and a constant recombination rate of 1.3×10^{-8} were used. We can clearly see from the simulation that long IBD segments tend to be very recent, whereas short IBD segments tend to be very old (Figure 1.7). On a 50 cM chromosome, the two individuals have 69 different IBD segments, with an average length of 0.73 cM (Figure 1.8). If we go deep enough into the past, every part of the chromosome meets a common ancestor between the two individuals. According to our results, the average most recent common ancestor connecting the two genomes lived around 35000 generations before present time. The

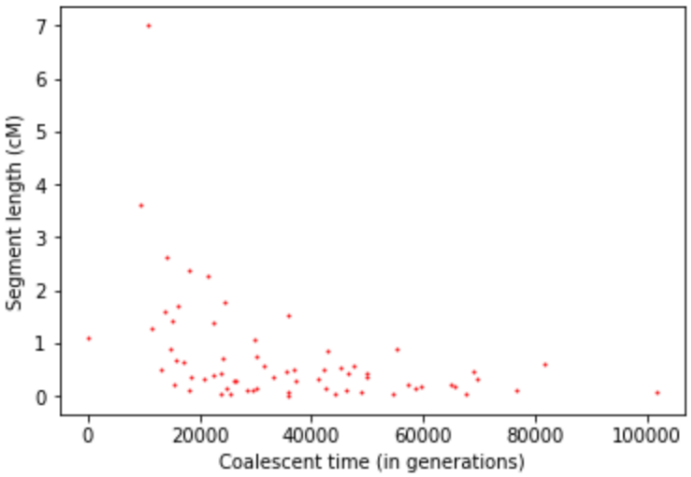


Figure 1.7: Length of two individuals' IBD segments in function of the time their MRCA simulated with Algorithm 2.

```
--------------------------------
Number of IBD segments : 69
Average length of IBD segments (in cM) :  0.728494562201
Standard deviation of IBD segments' length (in cM) :  1.04730551416
--------------------------------
Average coalescent time (in generation) :  35850
Standard deviation of coalescent time (in generation) :  20095.2517296
--------------------------------
Average recombination time (in generation) :  16306
Standard deviation of recombination time (in generation) :  12857.0526296
--------------------------------
Total number of mutations on the chromosome :  66
Average number of mutations per IBD segment :  1.54545454545
```

Figure 1.8: Statistics from the simulation based on Algorithm 2.

statistics distributions of the coalescent and recombination times are shown in Figure 1.9 and the distribution of the number of mutations per IBD is provided in Figure 1.10. We can see that most IBD segments don't have any mutated site. The Wright-Fisher model allowed us to understand better the notion of IBD segments through coalescent theory. We built our own IBD segments simulator and understood the statistics of the principal variables related to IBD segments.

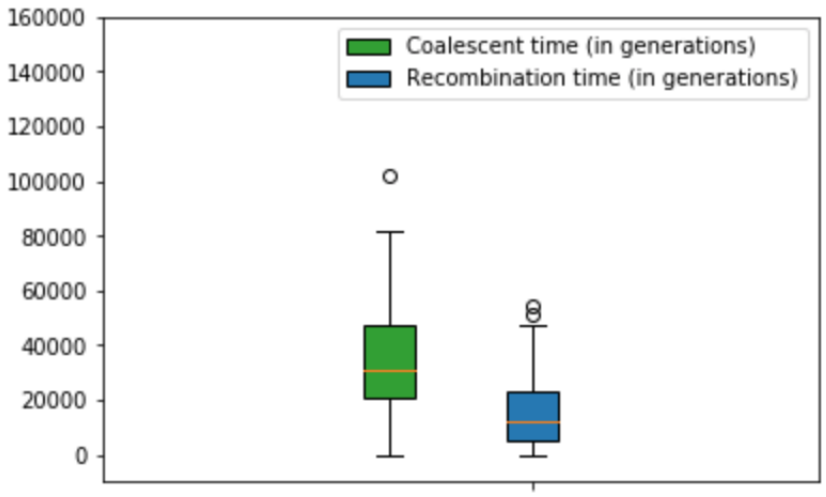


Figure 1.9: Coalescent and recombination times' distributions from the simulation.

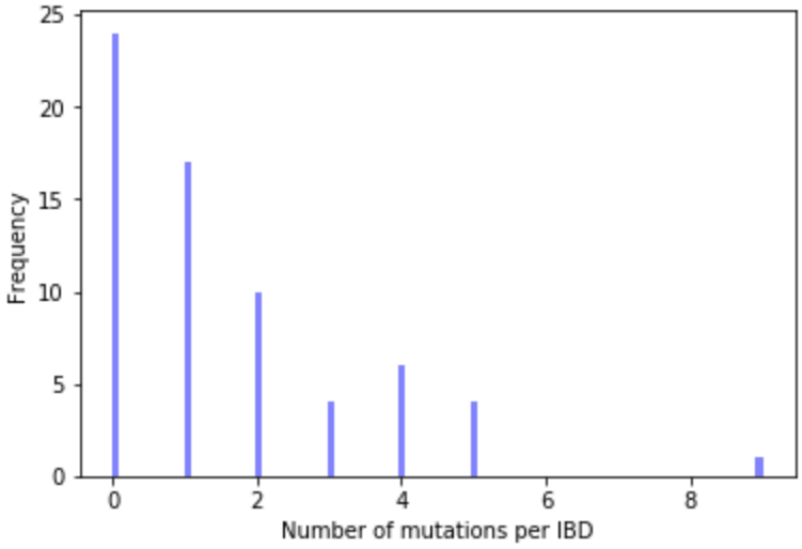


Figure 1.10: Frequency of mutations per IBD segment in the simulation.

Chapter 2

State of the art

There are many different existing algorithms to detect IBD segments. This chapter is an overview of the most recent and known methods for IBD detection. For more information, please directly refer to the corresponding paper. There are two principal kinds of methods for IBD detection: string-matching methods and model-based methods. We will go through these two types of methods, after introducing genomic data.

2.1 Overview of genomic data

Sequencing and array data Sequencing is a method used to determine the exact sequence of a certain length of DNA. This type of data is very expensive to get (around a few thousands of dollars per individual, see [24]). This is why we usually detect IBD segments on array data, which are a subset of sequence data with only the SNPs with a high frequency, and are much less expensive (from 10 to 100 dollars per sample, see [33]). Rare polymorphisms for which more than two alleles are present in the population are ignored and only biallelic polymorphisms are retained for our analysis.

Data format In the following, we will consider N diploid individuals ($2N$ haplotypes) and SNP array data. We will denote L the chromosome length in term of SNPs. Each of the $2N$ haplotypes is a binary sequence of L sites: 1 if the haplotype carries the minor allele of SNP, 0 otherwise. Genomic data refers to the genome and DNA data of an organism (human being or not) and are used to work on genomes.

Genotyping error We define genotyping as the process of determining the genetic variants of an individual to determine array data. The observed genotype of

an individual might be different from the true genotype. In this case, we say that genotyping errors occur. These errors affect most data and can lead to totally wrong conclusions. Previous works estimated the genotyping error rate per site depending on many different parameters. A genotyping error rate per site between 0.001 and 0.01 is commonly used [13]. This rate can sometimes be much higher. Four groups have been proposed to classify the different type of genotyping errors: errors linked to the DNA sequence itself, errors due to the low quality or quantity of the DNA, biochemical artefacts and human factors [30]. Let us see how negligible this error is. Let ε denote the genotyping error rate per site, and E_j the event of having a genotyping error on the j^{th} site. Note that for $i \neq j$, E_i and E_j are independent. Let E denote the event to have at least one site with a genotyping error in the whole sequence. This event is the complement of having no error at all in the whole sequence, which has probability $(1-\varepsilon)^L$, then:

$$\begin{aligned} P(E) &= 1 - (1-\varepsilon)^L \\ &= 1 - \exp\left(L \times \ln \varepsilon\right) \qquad with \quad 0 < \varepsilon < 1 \end{aligned}$$

Finally,

$$\lim_{L \to \infty} P(E) = 1$$

For a long chromosomal segment, each haplotype will be affected by genotyping error at least on one site, which will be always true in practice since $L >> 1$.

Phasing error When a method wants to output not only the individuals sharing IBD segments but also the exact haplotypes, it has to deal with phased data. Phased data distinguish the two haplotypes from the same individual, whereas unphased data only provide genotypes for each individual, without regard to which of the two homologous chromosomes carries the allele. A phasing error occurs when two alleles are switched from their original chromosome to the homologous one. In 2006, PHASE v2.1 was considered to be the most accurate algorithm for phasing inference with a phasing error of around 0.001 for related individuals and 0.05 for unrelated individuals [19].

2.2 String matching methods

2.2.1 Definition

String matching methods rely on string resemblance to detect IBD segments. These methods will only look for similar patterns on the chromosome for each possible pair of

individuals. Nevertheless, only relying on string resemblance has two main problems: first, mutations, genotyping errors and phasing errors can break string matched segments; secondly, individuals can have identical nucleotides on many successive sites without sharing any common ancestor (IBS and not IBD). Most string matching methods use some tricks to face the first issue, but the second one remains unsolved.

2.2.2 GERMLINE

GERMLINE is an algorithm released in 2009 [11]. We used the most recent available version, GERMLINE v.1.5.2.

Inputs GERMLINE takes as inputs a PED file, containing the phased data for all individuals, and a MAP file, providing the recombination rate for each site.

Parameters GERMLINE introduces three main parameters:

- the minimum length of an IBD segment min_m;
- the number of bits in each slice $bits$;
- the maximum number of mismatches allowed in each slice err.

Method GERMLINE divides the L SNPs of the $2N$ haplotypes in slices of $bits$ SNPs. For each of the $\lfloor \frac{L}{bits} \rfloor$ slices, GERMLINE builds a hash-table with constant-time insertion and lookup. For each hash-table, the key is a binary victor of length $bits$, corresponding to a portion of the chromosomal segment, and the value is a set of individual haplotypes sharing the key as part of their DNA in this region. While outputting the IBD segments, when two haplotypes happen to be in the same key's set for a slice, GERMLINE looks at the next slice to see if the two haplotypes still belong in the same key's set. If they do, GERMLINE keeps going, if they don't, the IBD segment breaks and is outputted if the total length is higher than min_m. Assuming no mutation, genotyping or phasing error, GERMLINE would just have to output all IBD segments right after processing the input. However, to face those issues, GERMLINE allows a maximum number of mismatches in each slice and merge all sets for which the keys have less than err mismatches.

Output One of GERMLINE's outputs is a file containing all IBD segments, each line corresponding to a pairwise shared segment. For each of them, GERMLINE provides its length and the total number of mismatches.

2.2.3 RaPID

RaPID was developped in 2017 [22]. Here, we used RaPID v.1.2 which is the most recent available version.

Inputs RaPID takes as an input a MaCS file, which contains information on all haplotypes and can be obtained from a VCF file.

Parameters RaPID has four principal parameters:

- the minimum length of IBD segments l;
- the window size w;
- the number of runs r;
- the minimum number of successes per site s.

Method RaPID uses the Positional Burrows-Wheeler Transform (PBWT) [9] to find matches between haplotypes efficiently. The basic idea of PBWT is to sort the haplotypes in order of reversed prefixes. By doing this, for a fixed position in the chromosome, all matches are neighbors, forming a block. Let us have a look at a small example for 3 haplotypes, $L = 6$ and positions $k \in \{1, ..., L\}$. We will underline the matches of each haplotype with its predecessor.

Haplotype 1: 010101
Haplotype 3: 011000
Haplotype 2: 111000

$k = 1$	$k = 2$	$k = 3$
010101	010101	010101
011000	011000	011000
111000	111000	111000

$k = 4$	$k = 5$	$k = 6$
011000	011000	011000
111000	111000	111000
010101	010101	010101

RaPID reads the whole dataset from the first position to the last one, and looks for IBD segments after each position among neighbors. However, to face mutations and both genotyping and phasing errors, RaPID divides the chromosome in windows of size w and makes r runs of PBWT. For each run, RaPID does a random projection in each window (it randomly selects a site in each window) and then runs PBWT on the randomly selected sites. RaPID merges all IBD segments between each run. Once all runs are completed, each IBD segment is outputted if and only if the segment has more than s successes, among the total number of attempts r.

Output RaPID outputs all pairwise shared segments found, with some characteristics such as the starting and ending positions.

2.3 Model-based methods

2.3.1 Overview

Model-based algorithms, on the other hand, rely on more complex notions than string matching and build a mathematical model to detect IBD segments.

2.3.1.1 Hidden Markov Model

Most of the time, model-based methods use a Hidden Markov Model (HMM).

Definition A HMM can be defined by four elements $\{S, \Pi, T, E\}$ where:

- S is the set of all possible (hidden) s states;
- Π is a vector of s elements, each π_i of them is the probability for S_i to be the initial (hidden) state;
- T is the $s \times s$ transition matrix, where T_{ij} is the transition probability from S_i to S_j;
- E is the $s \times e$ emission matrix, where E_{ij} is the probability to emit the j^{th} observation while being in the state S_i (where e is the total number of possible observations).

with:

$$\sum_{i=1}^{s} \pi_i = 1 \qquad \sum_{i=1}^{s} T_{ij} = 1 \qquad \sum_{j=1}^{s} E_{ij} = 1$$

A HMM is an inferential procedure based on dynamic programming in which we assume the system to be governed by a Markov process with unobserved hidden states (Figure 2.1). In the case of IBD segments detection, the states are usually some features related to an IBD segment (coalescent time for example) while the observations are the haplotypes of the pair of individuals considered. Different inference algorithms may then be used to predict the hidden states. We describe in the next section the two most commonly used algorithms to decode a HMM: the forward-backward algorithm and the Viterbi algorithm.

Notations Let X_t denote the random markovian variable representing all possible states at position t, for t in $[\![0, T]\!]$ where T is the total number of positions. Then:

$$\pi_i = P(X_0 = S_i)$$

and

$$P(X_t|X_{t-1}, X_{t-2}, ..., X_1, X_0) = P(X_t|X_{t-1})$$

Moreover,

$$T_{ij} = P(X_t = S_j|X_{t-1} = S_i)$$

We also denote the random variable representing the observation at position t by O_t. To simplify the next equations, we can introduce the diagonal $s \times s$ matrix $\mathcal{O}_j$ for the j^{th} observation such as:

$$(\mathcal{O}_j)_{ii} = P(O_t = j|X_t = i) = E_{ij}$$

The matrix $\mathcal{O}_i$ allows us to simplify the following:

$$P(O_t = j) = \sum_{i=1}^{s} \pi_i E_{ij} = \Pi \cdot \mathcal{O}_j$$

2.3.1.2 Forward-Backward Algorithm

The forward-backward algorithm is used to compute the posterior marginals of all hidden state variables given a sequence of observations and then choose the most likely state at each position. There are two steps in the forward-backward algorithm: on the one hand, it computes the forward probabilities while it computes the backward probabilities on the other hand.

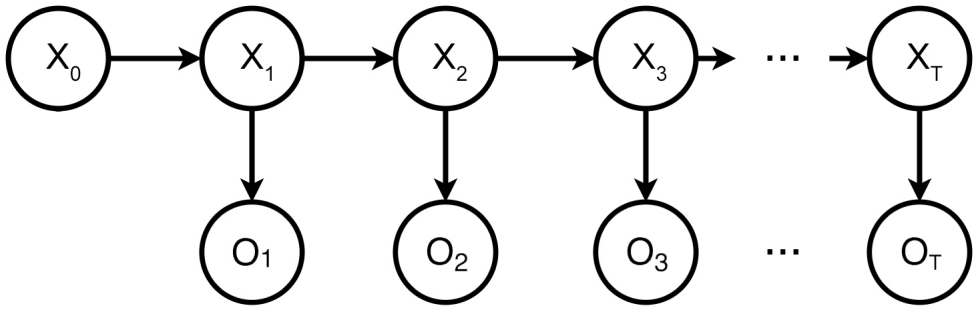


Figure 2.1: Hidden Markov Model where X_i is the hidden state and O_i is the observation at time step $t = i$. In the case of IBD segments detection, each time step represents a chromosomal position.

Forward probabilities Let $\mathcal{F}_t$ denote the vector of forward probabilities at time t such as: $(\mathcal{F}_t)_i = P(O_1, O_2, ..., O_t, X_t = i)$. Consequently,

$$\begin{aligned} \mathcal{F}_0 &= \pi_0 \\ \mathcal{F}_1 &= \pi_0 \cdot T \cdot \mathcal{O}_1 \\ &\cdots \\ \mathcal{F}_t &= \mathcal{F}_{t-1} \cdot T \cdot \mathcal{O}_t \end{aligned} \tag{2.1}$$

Backward probabilities Let $\mathcal{B}_t$ denote the vector of backward probabilities at time t such as: $(\mathcal{B}_t)_i = P(O_{t+1}, O_t, ..., O_T | X_t = i)$ with $\mathcal{B}_T = [1, 1, ..., 1]^T$. Consequently,

$$\mathcal{B}_t = \mathcal{T} \cdot \mathcal{O}_t \cdot \mathcal{B}_{t+1} \tag{2.2}$$

Posterior marginals of hidden states We can compute the posterior marginals of all hidden state variables by taking the product of the forward and backward probabilities:

$$\begin{aligned} (\mathcal{F}_t)_i \cdot (\mathcal{B}_t)_i &= P(O_1, O_2, ..., O_t, X_t = i) \cdot P(O_{t+1}, O_t, ..., O_T | X_t = i) && (2.3) \\ &= P(O_1, ..., O_t, X_t = i) \cdot P(O_{t+1}, O_t, ..., O_T | X_t = i, O_1, ..., O_t) && (2.4) \\ &= P(O_1, O_2, ..., O_T, X_t = i) && (2.5) \end{aligned}$$

Equation 3.4 follows from the conditional, independence of $O_{t+1}, O_t, ..., O_T$ and $O_1, ..., O_t$ given X_t. Finally, we define $c_t = P(O_t, O_{t-1}, ..., O_1) = \sum_{i=1}^{s} (\mathcal{F}_t)_i$ such as

$$P(X_t = i | O_1, O_2, ..., O_T) = \frac{P(O_1, O_2, ..., O_T, X_t = i)}{P(O_1, O_2, ..., O_T)} = \frac{(\mathcal{F}_t)_i \cdot (\mathcal{B}_t)_i}{c_T}$$

In the case of IBD detection, using the forward-backward algorithm allows us to predict the most likely state for each SNP for all observations of pairs of individuals.

2.3.1.3 Viterbi Algorithm

The Viterbi algorithm is used to find the most probable sequence of states $X_0, X_1, ..., X_T$ given a sequence of observations $O_1, O_2, ..., O_T$. The Viterbi algorithm is a kind of dynamic programming, making use of a trellis graph of s rows and T columns. It reads the sequence of observations from left to right and fills out the trellis as it goes along the sequence. Each cell is assigned a Viterbi score v_t^j corresponding to the probability of being in the j^{th} state given at position t the t^{th} first observations. The next Viterbi score is computed recursively considering the most likely path of states.

$$v_t^j = \max_{X_0, X_1, ..., X_{t-1}} P(O_1, O_2, ..., O_t, X_0, X_1, ..., X_{t-1}, X_t = j) \tag{2.6}$$

$$= \max_{i \in [\![1,s]\!]} (v_{t-1}^i \cdot T_{ij} \cdot E_{jt}) \tag{2.7}$$

with:

$$v_1^j = \max_{i \in [\![1,s]\!]} \pi_i \cdot E_{j1}$$

The Viterbi algorithm allows us to predict the most likely sequence of states for each observations of pairs of individuals in the case of IBD detection.

2.3.1.4 IBD score

Using a probabilistic model for IBD detection provides a very powerful advantage compared to string-matching techniques since it can output how confident the model is about the prediction. This is what we will call the IBD score. The higher the IBD score is, the more likely it is to be an IBD segment according to the model used. The IBD score will also be very important to compute the accuracy of the algorithm.

2.3.2 RefinedIBD

RefinedIBD was first released in 2013 in Beagle software package [2]. Here, we used the latest version released in July 2018.

Inputs RefinedIBD takes as an input a VCF file, containing the binary sequences for each haplotype and a MAP file, providing the recombination rate on each SNP.

Parameters RefinedIBD has two principal parameters:

- the minimum length in cM of an IBD segment *length*;
- the minimum LOD score to consider a segment as an IBD *lod*.

Method There are two main steps in the RefinedIBD algorithm. It first uses GERMLINE to generate candidate IBD segments, and then applies a probabilistic model to evaluate each of the segments.

Identification GERMLINE is first used with $bits = 64$ (corresponding to generally around 0.2 cM in SNP array data) and $err = 0$. GERMLINE's min_m parameter is controlled by RefinedIBD's *length* parameter. The candidate IBD segments are then sent to RefinedIBD's probabilistic model.

Refinement The RefinedIBD algorithm builds two haplotype frequency models: one for IBD and an other for non-IBD. It uses the Beagle HMM which determines a HMM for unrelated individuals and a HMM for parent-offspring individuals. Using a parent-offspring model to detect IBD between related individuals makes sense since parent-offspring pairs also share IBD segments. RefinedIBD computes the probability of each pair of observed haplotypes from the identification step using the two models: first, the likelihood of IBD using the HMM for parent-offspring individuals, and second, the likelihood of non-IBD using the HMM for unrelated individuals. Note that RefinedIBD only focuses on the candidate IBD segments found by the identification step which reduces the overall computational time. The likelihoods are computed using the forward-backward algorithm. Finally, RefinedIBD computes the LOD score for each candidate segment, which is the log likelihood ratio and only outputs segments having a LOD score higher than a threshold, specified by the user with the *lod* parameter. An empirical observation showed that the LOD score was highly impacted by non-IBD markers at the edges of the segment, and just slightly reduced when removing IBD markers of a real IBD segment. From this observation, RefinedIBD introduced the *ibdtrim* parameter to trim the candidate IBD segments on both edges. Its default value is 40 markers and is recommended to be around 0.15 cM. The trimmed markers are then restored to the IBD segment after computing the LOD score.

Output RefinedIBD outputs all IBD segments found, specifying the haplotypes, the starting and ending positions but unlike previous described methods, it provides the LOD score for each of them, which is an essential information for the user.

2.3.3 ASMC

The Ascertained Sequentially Markovian Coalescent (ASMC) algorithm was released in 2018 [29]. Here, we used the latest version.

Inputs ASMC takes as inputs a HAP file providing all the haplotypes, a MAP file with the recombination rate on each site and a Decoding Quantities file containing the time discretization and the Conditioned Sample Frequency Spectrum (CSFS) model containing the frequencies of each SNP. ASMC can be used on both SNP array and sequencing data.

Parameters ASMC has only one major parameter for IBD segments detection:

- the maximum time to consider the most recent common ancestor between individuals as an IBD t_{TMRCA} .

Method ASMC uses a HMM based on coalescent theory with a recombination process, which is derived from the pairwise sequentially Markovian coalescent (PSMC) approach [16]. Note that ASMC doesn't include gene conversion in its model. The time is discretized and divided into different intervals. The discretization has to be done before using ASMC. The hidden states of the HMM correspond to the time intervals, representing the coalescent time for a pair of individuals, also called the time of TMRCA. The states distribution initialization is done using the discretized coalescent density. The observations of the HMM are the frequencies of each SNP and the observed genotypes: for each pair of haplotypes, ASMC builds an observation variable, containing two binary vectors (XOR and AND sequences, Figure 2.2). The HMM emission relies on the conditional site frequency spectrum [34] modified to account for SNP ascertainment. The transitions in the HMM are based on the Sequentially Markovian Coalescent using the Conditional Simonsen-Churchill [14] transition model: they correspond to the changes in TMRCA. ASMC decodes the HMM using the forward-backward algorithm with a computational speed-up: the forward-backward run time is linear in number of SNPs and hidden HMM states (and not quadratic). To detect IBD, ASMC obtains Maximum-A-Posterior estimates (MAP)

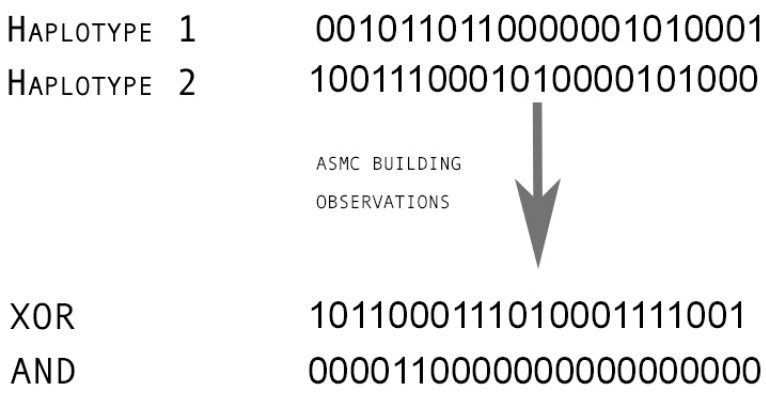


Figure 2.2: Building observation process in ASMC for each pair of haplotypes.

of TMRCA on each site: if the state for which the MAP is reached is smaller than t_{TMRCA}, it is considered to be part of an IBD segment, otherwise the segment breaks. To make the decoding process faster, ASMC uses batches of observations and run the forward-backward algorithm on multiple observations simultaneously. The number of observations in each batch can be modified by the user with the $batchSize$ parameter.

Output ASMC outputs all IBD segments having a common ancestor between present time and t_{TMRCA}. ASMC computes the mean MAP over the whole IBD segment which, as RefinedIBD's LOD score, is very valuable information for the user.. The average MAP is provided for each segment, with the starting and ending positions.

2.4 Comparison

We compare here the four methods described above. We first look at the accuracy and then the time complexity of each method.

2.4.1 Simulation of data

We simulated 10 independent datasets for a 30Mb chromosome and 300 haplotypes, using a model of European demographic history and recombination rates from a human chromosome 2. We mimicked SNP ascertainment by subsampling variants to match UK Biobank [4] allele frequencies. We first used ARGON simulator [28] to generate the data. We then used BCFtools [17] to convert HAP format files to VCF format and PLINK 1.9 to build MAP files [31].

2.4.2 Accuracy

2.4.2.1 Precision-Recall curve vs ROC curve

Detection of IBD can be seen as a binary classification problem. Each site of each observation is either an IBD or not. The class of IBD sites is the positive class, while the non-IBD class is the negative one. In the terminology true/false positive/negative, true or false refers to the assigned classification being correct or incorrect, while positive or negative refers to assignment to the positive or the negative category. Let TP denote the number of true positives, FP the number of false positives, TN the number of true negatives and FN the number of false negatives.

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.8}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.9}$$

$$\text{fall-out} = \frac{FP}{FP + TN} = 1 - \text{precision} \tag{2.10}$$

The precision-recall curve is made by plotting the precision against the recall at various threshold settings, while the receiver operating characteristic (ROC) shows the recall against the fall-out. For both plots, we compute the area under the curve. The higher it is, the better. Note that the maximum area under the curve is 1 for both metrics. However, IBD detection faces the class imbalance problem: the total number of data belonging in the IBD class (positive class) is much smaller than the total number of data in the non-IBD class (negative class). When a classification task faces the class imbalance problem, the precision-recall curve, which is insensitive to class imbalance, is more suitable. Let us have a look at an example to understand why. We consider a chromosome with 1000 IBD sites out of 1 million total sites and two methods to detect IBD. So here, we clearly have a class imbalance problem as the non-IBD class is 10^3 bigger than the IBD class. Using two different methods to detect IBD segments, we get:

Method 1: 1000 IBD sites detected, 900 real IBD sites.

$$TP = 900 \qquad FP = 100 \qquad TN = 999900 \qquad FN = 100$$

Consequently,

$$\text{precision} = \frac{900}{900 + 100} = 0.9$$

$$\text{recall} = \frac{900}{900 + 100} = 0.9$$

$$\text{fall-out} = \frac{100}{100 + 999900} = 0.0001$$

Method 2: 10000 IBD sites detected, 900 real IBD sites.

$$TP = 900 \qquad FP = 9100 \qquad TN = 990900 \qquad FN = 100$$

Consequently,

$$\text{precision} = \frac{900}{900 + 9100} = 0.09$$

$$\text{recall} = \frac{900}{900 + 100} = 0.9$$

$$\text{fall-out} = \frac{9100}{9100 + 990900} = 0.0091$$

We can clearly see here that looking only at the recall and the fall-out is not enough as there is almost no difference between both methods. We can imagine similar results no matter the threshold used for both methods, and we won't be able to compare them. On the contrary, as the precision doesn't consider true negatives, it is not affected by the class imbalance and reveals a big difference between both methods.

2.4.2.2 Thresholds

We need thresholds for all methods to plot their precision-recall curves. For the model-based methods, the threshold is explicitly given: the LOD score for RefinedIBD, and the average probability for ASMC. However, for the string matching techniques, we had to find a variable that we could use as IBD score. For both RaPID and GERMLINE, based on empirical observations, we decided to take the segment's length as IBD score. The length makes sense since long segments are more likely to be recent and not affected by mutations (see Chapter 1). The higher the length is, the more likely it is to be an IBD segment.

2.4.2.3 IBD definition

We defined a time threshold t such as a region is considered to be IBD if the most recent common ancestor for a pair of individuals is less than t generations. To test IBD detection accuracy at different time scales, we compared the four algorithms' ability to detect IBD within the past 25, 50, 100, 250, 400 and 600 generations.

2.4.2.4 Grid search

We used a grid-search over possible parameters values to optimize the accuracy of all methods, measured using area under the precision-recall curve for all time thresholds mentioned above, and the time complexity at the same time. The grid search was done on a different dataset, independent of the 10 simulated datasets used for benchmarking. We provide here the best sets of parameters found for each method.

GERMLINE

t	25	50	100	250	400	600
min_m	0.001	0.001	0.001	0.001	0.001	0.001
$bits$	128	128	128	64	50	50
err	2	5	10	10	10	10

RaPID

t	25; 50; 100	250; 400; 600
l	0.5	0.5
r	10	10
s	8	5
w	3	3

RefinedIBD

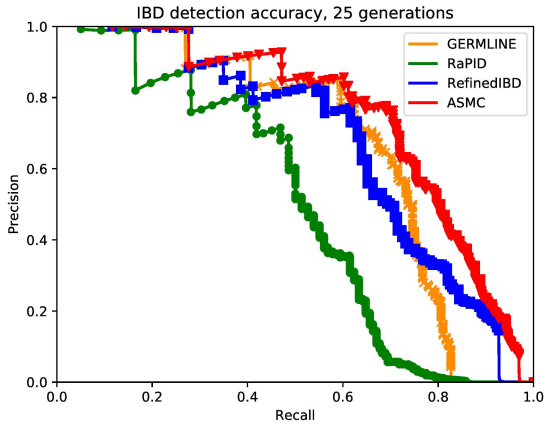
t	25	50	100	250	400	600
$length$	1.5	1	0.75	0.01	0.001	0.001
lod	0.001	0.001	0.001	0.001	0.001	0.001

ASMC The time parameter of ASMC was chosen to be the same as the time threshold used for accuracy.

2.4.2.5 Results

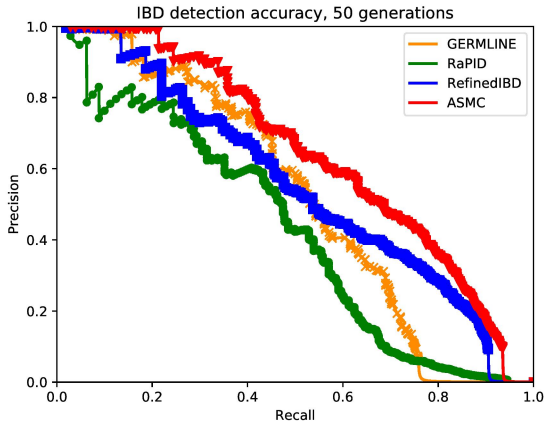
We computed the average recall range and area under the precision recall curve (auPRC) for each method and each time threshold. Numbers in round brackets represent standard errors on the 10 simulations. For each time threshold, we randomly sample one dataset out of the 10 and show the corresponding precision recall plot. See Figures 2.3, 2.4, 2.5, 2.6, 2.7 and 2.8. We can see from the results that the deeper in time we focus on, the more model-based methods tend to achieve better results. As we saw in Chapter 1, the deeper in time we go, the shorter IBD segments tend to be. String-matching techniques cannot detect too short segments, as they get very easily confused with IBS segments and miss segments with mutations. RaPID

and GERMLINE both have parameters to deal with segments with mutations though. However, there is a trade-off between allowing segments with mutations, which means being more permissive, and considering false positive segments. We also notice that for all methods, the recall is never defined from 0 to 1. If we go back to the definition of recall (Equation 3.11), we can see that recall provides information about how many IBD segments are being missed. This means that we always miss some real IBD segments for each method. The restricted recall range for RefinedIBD is due to the LOD parameter. For ASMC, the time discretization is one of the most important parameter explaining the recall range. Because of the minimum length parameter introduced in string-matching techniques, RaPID and GERMLINE will always miss segments shorter than the length threshold, which is a major disadvantage of string-matching techniques.



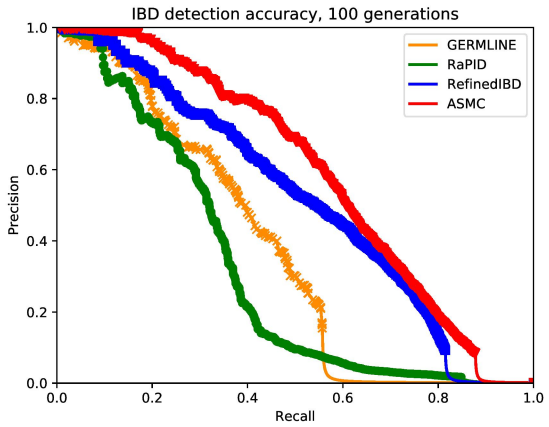
method	auPRC	recall range
GERMLINE	0.56 (0.02)	[0.07, 0.83]
RaPID	0.47 (0.02)	[0.06, 0.97]
RefinedIBD	0.57 (0.02)	[0.06, 0.92]
ASMC	0.64 (0.02)	[0.07, 0.97]

Figure 2.3: Accuracy results obtained for IBD detection within the past 25 generations on 10 independent datasets.



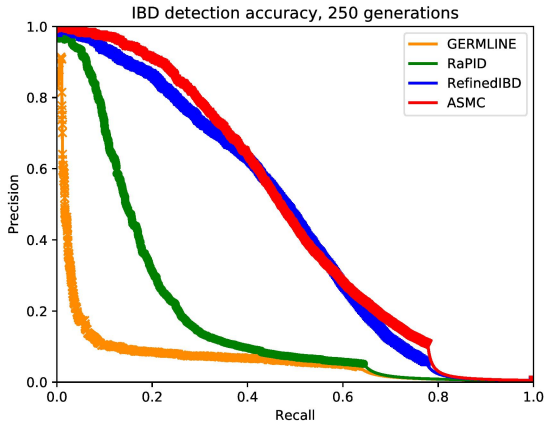
method	auPRC	recall range
GERMLINE	0.51 (0.01)	[0.03, 0.73]
RaPID	0.41 (0.01)	[0.03, 0.73]
RefinedIBD	0.55 (0.01)	[0.03, 0.89]
ASMC	0.63 (0.01)	[0.03, 0.92]

Figure 2.4: Accuracy results obtained for IBD detection within the past 50 generations on 10 independent datasets.



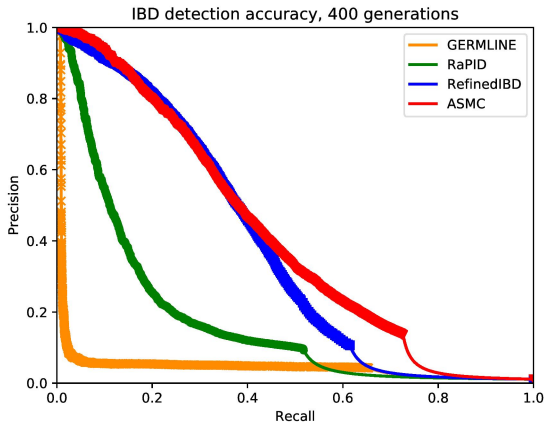
method	auPRC	recall range
GERMLINE	0.33 (0.01)	[0.01, 0.57]
RaPID	0.29 (0.01)	[0.01, 0.85]
RefinedIBD	0.51 (0.01)	[0.01, 0.81]
ASMC	0.58 (0.01)	[0.01, 0.86]

Figure 2.5: Accuracy results obtained for IBD detection within the past 100 generations on 10 independent datasets.



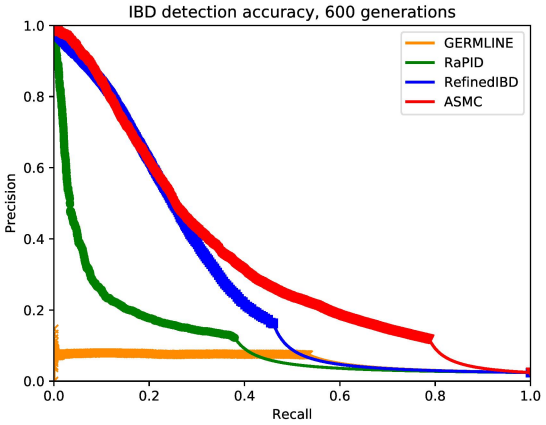
method	auPRC	recall range
GERMLINE	0.07 (0.00)	[0.00, 0.63]
RaPID	0.17 (0.01)	[0.00, 0.63]
RefinedIBD	0.43 (0.00)	[0.00, 0.67]
ASMC	0.46 (0.00)	[0.00, 0.77]

Figure 2.6: Accuracy results obtained for IBD detection within the past 250 generations on 10 independent datasets.



method	auPRC	recall range
GERMLINE	0.04 (0.00)	[0.00, 0.64]
RaPID	0.14 (0.01)	[0.00, 0.50]
RefinedIBD	0.37 (0.00)	[0.00, 0.63]
ASMC	0.39 (0.01)	[0.00, 0.74]

Figure 2.7: Accuracy results obtained for IBD detection within the past 400 generations on 10 independent datasets.



method	**auPRC**	**recall range**
GERMLINE	0.06 (0.00)	[0.00, 0.53]
RaPID	0.13 (0.01)	[0.00, 0.38]
RefinedIBD	0.29 (0.00)	[0.00, 0.46]
ASMC	0.36 (0.01)	[0.00, 0.78]

Figure 2.8: Accuracy results obtained for IBD detection within the past 600 generations on 10 independent datasets.

2.4.3 Time complexity

To compare the time complexity of all methods, we simulated six different datasets of 150, 300, 450, 750, 1500 and 2500 diploid individuals, for a 30Mb chromosome, using a model of European demographic history and recombination rates from a human chromosome 2. As before, we mimicked SNP ascertainment by subsampling variants to match UK Biobank allele frequencies. Because we used different sets of parameters depending on the time threshold t_{IBD} (25, 50, 100, 250, 400 and 600 generations), we compared the time complexity for each of them. Note that only one thread has been used for each algorithm in order to get a fair comparison (no multi threading or any other parallelism techniques). We can see from the plots (see Figure 2.9) that model-based methods don't scale as well as string-matching techniques. RaPID and GERMLINE are almost linear in time while RefinedIBD and ASMC are quadratic. We can note that for time thresholds close to present time (below 50 generations), RefinedIBD scales very well though. This is due to the fact that for small time thresholds, IBD segments are generally very long, so the identification step of RefinedIBD is done with a high minimum length parameter, which makes RefinedIBD' s time complexity close to GERMLINE's, which is the algorithm used by RefinedIBD for the identification step.

2.4.4 The accuracy-scalability trade-off

There definitely is a trade-off between accuracy and scalability in IBD detection. In practice, we would like to detect IBD segments among hundred of thousand individuals but also reach the best accuracy possible. Algorithms that scale to data sets comprising hundreds of thousands of samples such as GERMLINE and RaPID favor computational speed over IBD detection accuracy. On the other hand, model-based algorithms such as RefinedIBD and ASMC cannot scale to very large data sets.

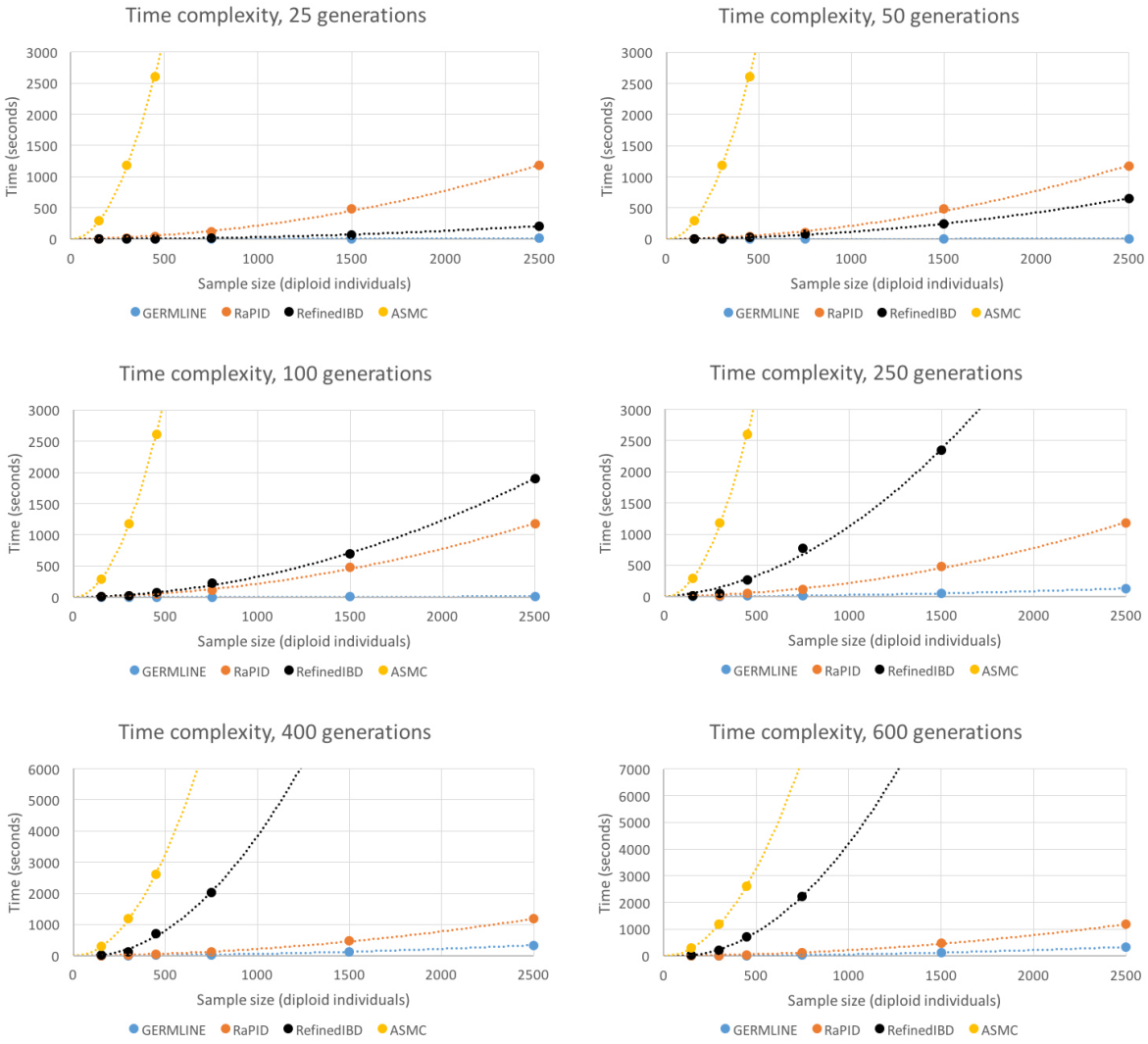


Figure 2.9: Time complexity for every IBD time threshold considered.

Chapter 3

Preprocessed Ascertained Sequentially Markovian Coalescent

3.1 Method

We present here the Preprocessed Ascertained Sequentially Markovian Coalescent (PASMC) algorithm for IBD detection. There are two different steps in our method. The PBWT algorithm is first applied multiple times to the dataset with random projections. This preliminary work results in IBD candidates with high sensitivity and low specificity. These candidates are then filtered using the ASMC method. PASMC has been built in C++.

3.1.1 Identification of IBD candidates

The identification of IBD candidates is based on RaPID algorithm described in Section 2.2.3.

3.1.1.1 Positional Burrows-Wheeler transform (PBWT)

We implemented the PBWT algorithm, which sorts the haplotypes in order of reversed prefixes (see Chapter 2 for more details). Richard Durbin described the algorithm in 2014 [9] and we present here the main steps of the method. Please refer to [21] for a Python implementation of PBWT.

Building the positional prefix array Durbin introduced the "positional prefix array", which is the list of haplotype indices that sorts the haplotypes in reversed prefix order. The positional prefix array is defined for each position and built recursively from the previous position's positional prefix array, which allows respecting the

previous order when the same allele is carried by different haplotypes. The prefix array is initialized following the original order of the haplotypes. See Algorithm 3.

Algorithm 3 Positional prefix array

INPUT: X - matrix of haplotypes; K - position
OUTPUT: PPA - positional prefix array

```
1: N ← number of rows of X (number of haplotypes)
2: L ← number of columns of X (number of SNPs)
3: PPA ← new List[]  list of size L containing all integers from 1 to L
4: for k in [[1, K]] do
5:     hapMaj ← new List[]  haplotypes' indexes carrying the major allele 0 at k
6:     hapMin ← new List[]  haplotypes' indexes carrying the minor allele 1 at k
7:     for index in PPA do
8:         hap ← X[index]
9:         allele ← hap[k]
10:        if allele== 0 then hapMaj.append(index)
11:        else hapMin.append(index)
12:    PPA ← hapMaj+hapMin
   return PPA
```

Building the divergence array Durbin also introduced the "divergence array" at each position, which is the list of the starting positions of matches between all haplotypes and their predecessors. So for a fixed position k, the i^{th} element of the divergence array is the starting position of the match between the i^{th} haplotype in the sorted order (i^{th} element of the prefix array) and its predecessor. The divergence array is based on the fact that the match between two haplotypes didn't break as long as they remain adjacent in the sorted order. The divergence array is initialized with 1 everywhere. At position k, the divergence of the two haplotypes carrying the first observed minor and major alleles is $k + 1$ as they don't share any segment with their predecessors. See Algorithm 4.

Algorithm 4 Divergence array

INPUT: X - matrix of haplotypes; K - position

OUTPUT: PPA - positional prefix array at K; DIV - divergence array at K

$N \leftarrow$ number of rows of X (number of haplotypes)
$L \leftarrow$ number of columns of X (number of SNPs)
$PPA \leftarrow$ *new List[]* list of size L containing all integers from 1 to L
$DIV \leftarrow$ *new List[]* list of size L full of 1
for k in $[\![1, K]\!]$ **do**
    $hapMaj \leftarrow$ *new List[]* haplotypes' indexes carrying the major allele 0 at k
    $divMaj \leftarrow$ *new List[]* div array for haplotypes carrying the major allele 0 at k
    $hapMin \leftarrow$ *new List[]* haplotypes' indexes carrying the minor allele 1 at k
    $divMin \leftarrow$ *new List[]* div array for haplotypes carrying the minor allele 1 at k
    $divMaj_k, divMin_k \leftarrow k+1$ initializing the starting match position
    for index, matchStart in PPA, DIV **do**
        $hap \leftarrow X[index]$
        $allele \leftarrow hap[k]$
        if allele== 0 **then**
            $hapMaj.append(index)$
            $divMaj.append(divMaj_k)$
            $divMaj \leftarrow 1$
        else
            $hapMin.append(index)$
            $divMin.append(divMin_k)$
            $divMin \leftarrow 1$
    $PPA \leftarrow hapMaj+hapMin$
    $DIV \leftarrow divMaj+divMin$
return PPA, DIV

Reporting long matches We now want to report all matches longer than a length threshold L_{IBD}. The PBWT algorithm allowed us to sort all haplotypes in a way that all haplotypes with matches longer than a certain threshold are adjacent. These adjacent haplotypes form a block. Thus, we just need to iterate over all haplotypes in the sorted order, and report all haplotypes with matches longer than L_{IBD}. When a match between two adjacent haplotypes is smaller than L_{IBD}, it means a break in the block. We introduce two lists, $blockMaj$ and $blockMin$, containing the haplotypes' indexes with respectively the major and minor allele at position k. See Algorithm 5. Algorithm 5 outputs a list of matches with the format k, $blockMaj$, $blockMin$, corresponding to all matches ending at position k, between haplotypes from blockMaj list and blockMin list. Note that there is one problem using PBWT algorithm. Matches are only reported if they break at some point, so matches that include the last position won't be reported. This problem is solved with a boolean condition in PASMC code.

Algorithm 5 Reporting long matches

INPUT: X - matrix of haplotypes; K - position; L_{IBD} - length threshold

```
1: N ← number of rows of X (number of haplotypes)
2: L ← number of columns of X (number of SNPs)
3: PPA ← new List[]  list of size L containing all integers from 1 to L
4: DIV ← new List[]  list of size L full of 1
5: for k in ⟦1, K⟧ do
6:     hapMaj ← new List[]  haplotypes' indexes carrying the major allele 0 at k
7:     divMaj ← new List[]  div array for haplotypes carrying the major allele 0 at k
8:     blockMaj ← new List[]  block of haplotypes carrying the major allele 0 at k
9:     hapMin ← new List[]  haplotypes' indexes carrying the minor allele 1 at k
10:    divMin ← new List[]  div array for haplotypes carrying the minor allele 1 at k
11:    blockMin ← new List[]  block of haplotypes carrying the minor allele 1 at k
12:    divMaj_k, divMin_k ← k+1  initializing the starting match position
13:    for index, matchStart in PPA, DIV do
14:        if matchStart> k − L_IBD then
15:            if blockMaj and blockMin not empty then
16:                print k, blockMaj, blockMin
17:            blockMaj ← new List[]
18:            blockMin ← new List[]
19:        hap ← X[index]
20:        allele ← hap[k]
21:        if matchStart>divMaj then
22:            divMaj_k ← matchStart
23:        if matchStart>divMin then
24:            divMin_k ← matchStart
25:        if allele== 0 then
26:            hapMaj.append(index)
27:            divMaj.append(divMaj_k)
28:            divMaj ← 1
29:            blockMaj.append(index)
30:        else
31:            hapMin.append(index)
32:            divMin.append(divMin_k)
33:            divMin ← 1
34:            blockMin.append(index)
35:    if blockMaj and blockMin not empty then
36:        print k, blockMaj, blockMin
37:    PPA ← hapMaj+hapMin
38:    DIV ← divMaj+divMin
```

Reporting starting position For the purpose of IBD segments detection, we modified PBWT algorithm to report not only the ending position of a match, but also the starting position. We noticed that the starting position of a match between two haplotypes is the maximum starting position between these two haplotypes in the sorted order. So at position k, the starting position of the match between the i^{th} and j^{th} haplotype (with $i < j$) is the maximum element in the divergence array between $i+1$ and j. See Algorithm 6 for complete string-matching detection. Algorithm 6 outputs all IBD candidate segments found by the string-matching method PBWT. Each line is an IBD candidate segment, with the inclusive starting and ending positions and the pair of haplotypes.

Algorithm 6 String-matching detection

INPUT: X - matrix of haplotypes; K - position; L_{IBD} - length threshold

$N \leftarrow$ number of rows of X (number of haplotypes)
$L \leftarrow$ number of columns of X (number of SNPs)
$PPA \leftarrow$ *new List[]* list of size L containing all integers from 1 to L
$DIV \leftarrow$ *new List[]* list of size L full of 1
for k in $[\![1, K]\!]$ **do**
    $hapMaj \leftarrow$ *new List[]* haplotypes' indexes carrying the major allele 0 at k
    $divMaj \leftarrow$ *new List[]* div array for haplotypes carrying the major allele 0 at k
    $blockMaj \leftarrow$ *new List[]* block of haplotypes carrying the major allele 0 at k
    $hapMin \leftarrow$ *new List[]* haplotypes' indexes carrying the minor allele 1 at k
    $divMin \leftarrow$ *new List[]* div array for haplotypes carrying the minor allele 1 at k
    $blockMin \leftarrow$ *new List[]* block of haplotypes carrying the minor allele 1 at k
    $divMaj_k, divMin_k \leftarrow k+1$ initializing the starting match position
    for index, matchStart in PPA, DIV **do**
        if matchStart$> k - L_{IBD}$ **then**
            if blockMaj and blockMin not empty **then**
                for hap1 in blockMaj **do**
                    for hap2 in blockMin **do**
                        $index1 \leftarrow$ get position of hap1 in PPA
                        $index2 \leftarrow$ get position of hap2 in PPA
                        $index3 \leftarrow$ min(index1,index2)
                        $index4 \leftarrow$ max(index1,index2)
                        $startingPosition \leftarrow$ max element of DIV between index3+1 and index4
                        print $startingPosition, k-1, hap1, hap2$
            $blockMaj \leftarrow$ *new List[]*
            $blockMin \leftarrow$ *new List[]*

```
26:         hap ← X[index]
27:         allele ← hap[k]
28:         if matchStart>divMaj then
29:             divMaj_k ← matchStart
30:         if matchStart>divMin then
31:             divMin_k ← matchStart
32:         if allele== 0 then
33:             hapMaj.append(index)
34:             divMaj.append(divMaj_k)
35:             divMaj ← 1
36:             blockMaj.append(index)
37:         else
38:             hapMin.append(index)
39:             divMin.append(divMin_k)
40:             divMin ← 1
41:             blockMin.append(index)
42:     if blockMaj and blockMin not empty then
43:         for hap1 in blockMaj do
44:             for hap2 in blockMin do
45:                 index1 ← get position of hap1 in PPA
46:                 index2 ← get position of hap2 in PPA
47:                 index3 ← min(index1,index2)
48:                 index4 ← max(index1,index2)
49:                 startingPosition ← max element of DIV between index3+1 and index4
50:                 print startingPosition, k − 1, hap1, hap2
51:     PPA ← hapMaj+hapMin
52:     DIV ← divMaj+divMin
```

3.1.1.2 Multiple runs with random projections

To face mutations, genotyping and phasing errors, we do multiple runs of PBWT with random projections. We denote the number of runs by r. We define a window size w to divide the chromosome in equal slices. For each window, the projected site has a uniform distribution on the interval $[\![1, w]\!]$. The PBWT algorithm is done on the projected sites only and outputs a list of IBD candidate segments longer than a length threshold L_{IBD}. We repeat the operation r times while merging the IBD segments. Once all the runs are completed, if an IBD candidate segment has a number of successes smaller than the minimum number of successes s, it is finally deleted. An IBD candidate segment is a set of integers $(h1, h2, a, b, s)$, where $h1$ and $h2$ are respectively the first and second haplotype's index with $h1 < h2$, a is the starting position, b is the ending position and s is the number of successes. The number of successes is initialized to 1.

Definition 3.1.1. *Two segments* $(h1, h2, a_1, b_1, s_1)$ *and* $(h3, h4, a_2, b_b, s_2)$ *overlap if and only if all the following conditions are met:*

- $h1 = h3$
- $h2 = h4$
- $a_1 \leq b_2$
- $a_2 \leq b_1$

Through all the runs, when two segments overlap, we merge them by taking the union of the two segments. For a fixed pair of haplotypes $(h1, h2)$, two segments (a_1, b_1, s_1) and (a_2, b_b, s_2) are merged and transformed into one new segment $(h1, h2, a, b, s)$ where:

$$a = \min(a_1, a_2)$$

$$b = \max(b_1, b_2)$$

$$s = s_1 + s_2$$

We decided to merge segments taking the union to be sure we don't miss any IBD site. We want the identification step to detect IBD candidate with maximum recall. We also tried various methods of merging, such as taking the average of all overlapping segments' starting and ending positions, but the best results we obtained were using the minimum and maximum functions.

3.1.2 Validation of IBD candidates

The validation of IBD candidates is based on ASMC algorithm described in section 3.3.3.

Ascertained Sequentially Markovian Coalescent (ASMC) We directly had access to ASMC source code. The idea here was to decode the HMM only for the chromosomal regions and pairs of individuals of interest, rather than doing it on the whole chromosome for each possible pair of individuals. That would make ASMC much faster. The challenge here was to obtain the same accuracy as ASMC alone.

Building observations Based on the list of IBD candidates provided by the identification step, we now build observations only for the pairs of individuals and chromosomal regions which are likely to be IBD (see Figure 3.1). However, because a HMM needs some time steps to stabilize, we decided to add 0.5 cM on both edges of each candidate segment's chromosomal region.

Definition of an IBD segment Because we use a string-matching to identify the candidates, we faced the problem of the small recall range (see Chapter 2). To increase the recall range, we modified the definition of an IBD segment used by ASMC algorithm. ASMC computes the MAP of TMRCA on each site and considers a site as an IBD if the state reaching the MAP is smaller than the time threshold t_{TMRCA}. A small recall range means that we are missing too many true IBD segments, so we need to make PASMC more permissive. To do that, we introduce a new definition of an IBD segment. On each site, we compute the sum of all posterior marginals between present time and t_{TMRCA}. If this sum is higher than $\frac{k}{s}$ where s is the total number of states and k is the number of states between present time and t_{TMRCA} ($\frac{k}{s}$ is the uniform probability), the site is considered to be an IBD. In other words, if our belief that the considered site is an IBD is stronger than a random guess, we finally consider it as an IBD, otherwise, the segment breaks. The IBD score of a segment is the average sum of all posterior marginals between present time and t_{TMRCA} on each site of the segment. This new definition of an IBD site makes PASMC more permissive and increased considerably the recall range.

3.1.3 Computational speed-ups

We used many tricks to optimize the algorithm.



Figure 3.1: Building observations in ASMC vs PASMC for three different haplotypes. x represents a SNP not identified as IBD by PASMC. In this example, the minimum length is 5 SNPs, the window size is 1, the number of runs is 1 and the minimum number of successes is 1. ASMC considers all $\binom{3}{2}$ pairs of individuals and the whole chromosomal segment for each of them, while PASMC builds observations only on IBD candidate segments.

Automatic vectorization We used auto-vectorization while compiling PASMC to transform sequential parts of the algorithm into equivalent parallel ones. For example, the compiler automatically looks for loops that can be vectorized, relying on data dependencies inside loops, and performs loop vectorization. This can lead to significant automatic performance gains, especially on large data sets.

Batches ASMC doesn't decode observations one by one. It uses batches of observations, with a batch size parameter that can be modified by the user. This allows the program to do the forward-backward algorithm only once per batch, instead of once per observation. We decided do the same with PASMC but now that all observations have different starting and ending positions, we decided to decode each batch on the union chromosomal region of all segments. To do so, when a batch is full (the default batch size parameter for PASMC is 16), we compute the minimum of all starting positions and the maximum of all ending positions. We then do the forward-backward algorithm for all observations in the new common chromosomal region.

Sorting IBD candidates We should minimize as possible the number of extra sites we add for each of the observations because these extra sites were not identified as IBD in the identification step. To do that, before sending all IBD candidate segments to the verification step, we sort them by starting position, and then by ending position. This allows us to have batches of observations from the same chromosomal region.

Merging non overlapping segments Remember that we add 0.5 cM on both edges of each IBD candidate segment, which means that for a fixed pair of haplotypes, if we have two IBD segments (a_1, b_1, s_1) and (a_2, b_2, s_2) with $b_1 < a_2$, we will decode the first one from $a_1 - 0.5$ to $b_1 + 0.5$ and the second one from $a_2 - 0.5$ to $b_2 + 0.5$. If only a few sites separate the two segments (one site in the worst case), we will have $a_2 - 0.5\text{cM} < b_1 + 0.5$ and decode twice the same chromosomal region. Moreover, edges are less likely to be true positives since we manually add them and there were not identified as IBD by the multiple runs of PBWT. Consequently, there is an impact on the time complexity of the whole algorithm but also on the accuracy, as the verification step might consider the overlapping segment as an IBD twice. To avoid this issue, we decided to introduce a new variable called *step* when we merge segments. The *step* is automatically initialized in the algorithm depending on the value of the minimum IBD segment length parameter. We now have a new definition of two overlapping segments and merge them the same way as before.

Definition 3.1.2. *Two segments* $(h1, h2, a_1, b_1, s_1)$ *and* $(h3, h4, a_2, b_b, s_2)$ *overlap if and only if all the following conditions are met:*

- $h1 = h3$
- $h2 = h4$
- $a_1 - step \leq b_2$
- $a_2 \leq b_1 + step$

3.1.4 Parameters

List of parameters PASMC has different parameters, here are the most important ones:

- the minimum length of IBD segments in term of SNPs l;
- the window size in term of SNPs w;
- the number of runs r;
- the minimum number of successes s;
- the maximum number of generations to consider the most recent common ancestor between individuals as an IBD t_{TMRCA}.

Grid search As previously done for other methods, we did a grid search to find the best set of parameters for each time threshold t. Remember that the grid search was performed on an independent training dataset to both maximize the area under the precision-recall curve for all considered time thresholds and to minimize the time complexity.

$\boldsymbol{t}$	**25**	**50**	**100**	**250**	**400**	**600**
$\boldsymbol{t_{TMRCA}}$	25	50	100	250	400	600
$\boldsymbol{l}$	250	230	200	170	140	100
$\boldsymbol{w}$	5	5	5	5	5	5
$\boldsymbol{r}$	10	10	10	10	10	10
$\boldsymbol{s}$	8	8	8	8	8	8

We also did a grid search over the *step* parameter and here are the best values of the parameter we found for the corresponding length l parameter. Note that the *step* parameter is automatically initialized for the user. Values of the *step* parameter are given in cM while the values of the *length* parameter are given in number of SNPs.

$\boldsymbol{l}$	250	230	200	170	140	100
step	15	10	2	1	0.5	0.5

3.2 Benchmarking

3.2.1 Accuracy

We compared PASMC's accuracy with other algorithms using the percent auPRC improvement defined below. We used the same 10 independent datasets we simulated and described in Section 2.4.1.

Definition 3.2.1. *Given two methods M1 and M2 and their auPRC denoted respectively by* $auPRC1$ *and* $auPRC2$*, we define the percent auPRC improvement of M1 over M2 as:*

$$\frac{auPRC1 - auPRC2}{auPRC2} \times 100$$

Note that to get a fair comparison between all algorithms, we computed the percent auPRC improvement on the common recall range, as they all have a different recall range (see Section 2.4.2.5). We focused our analysis only on the following time thresholds: 100, 250 and 400 generations. Numbers in round brackets represent standard errors on the 10 simulations. We present in Figures 3.2, 3.3 and 3.4 the obtained results. PASMC's accuracy is very similar to ASMC's accuracy, no matter the time threshold. It also outperforms every other existing methods. We notice that for very large time threshold, RefinedIBD achieves similar results though.

method	auPRC within recall range	percent auPRC improvement
PASMC	0.41 (0.01)	
GERMLINE	0.38 (0.01)	8.94 (1.01)
RaPID	0.31 (0.01)	40.51 (3.61)
RefinedIBD	0.38 (0.01)	8.85 (1.51)
ASMC	0.46 (0.01)	-4.95 (0.62)

Figure 3.2: Accuracy results obtained with an IBD time threshold of 100 generations on 10 independent datasets. The average common recall range is [0.01, 0.50].

method	auPRC within recall range	percent auPRC improvement
PASMC	0.24 (0.00)	
GERMLINE	0.20 (0.00)	16.07 (0.96)
RaPID	0.15 (0.01)	56.37 (4.96)
RefinedIBD	0.23 (0.00)	2.76 (0.36)
ASMC	0.24 (0.00)	-1.73 (0.18)

Figure 3.3: Accuracy results obtained with an IBD time threshold of 250 generations on 10 independent datasets. The average common recall range is [0.00, 0.26].

method	auPRC within recall range	percent auPRC improvement
PASMC	0.15 (0.00)	
GERMLINE	0.13 (0.00)	15.82 (1.40)
RaPID	0.08 (0.00)	83.48 (9.36)
RefinedIBD	0.15 (0.00)	0.65 (0.27)
ASMC	0.15 (0.00)	-1.07 (0.11)

Figure 3.4: Accuracy results obtained with an IBD time threshold of 400 generations on 10 independent datasets. The average common recall range is [0.00, 0.16].

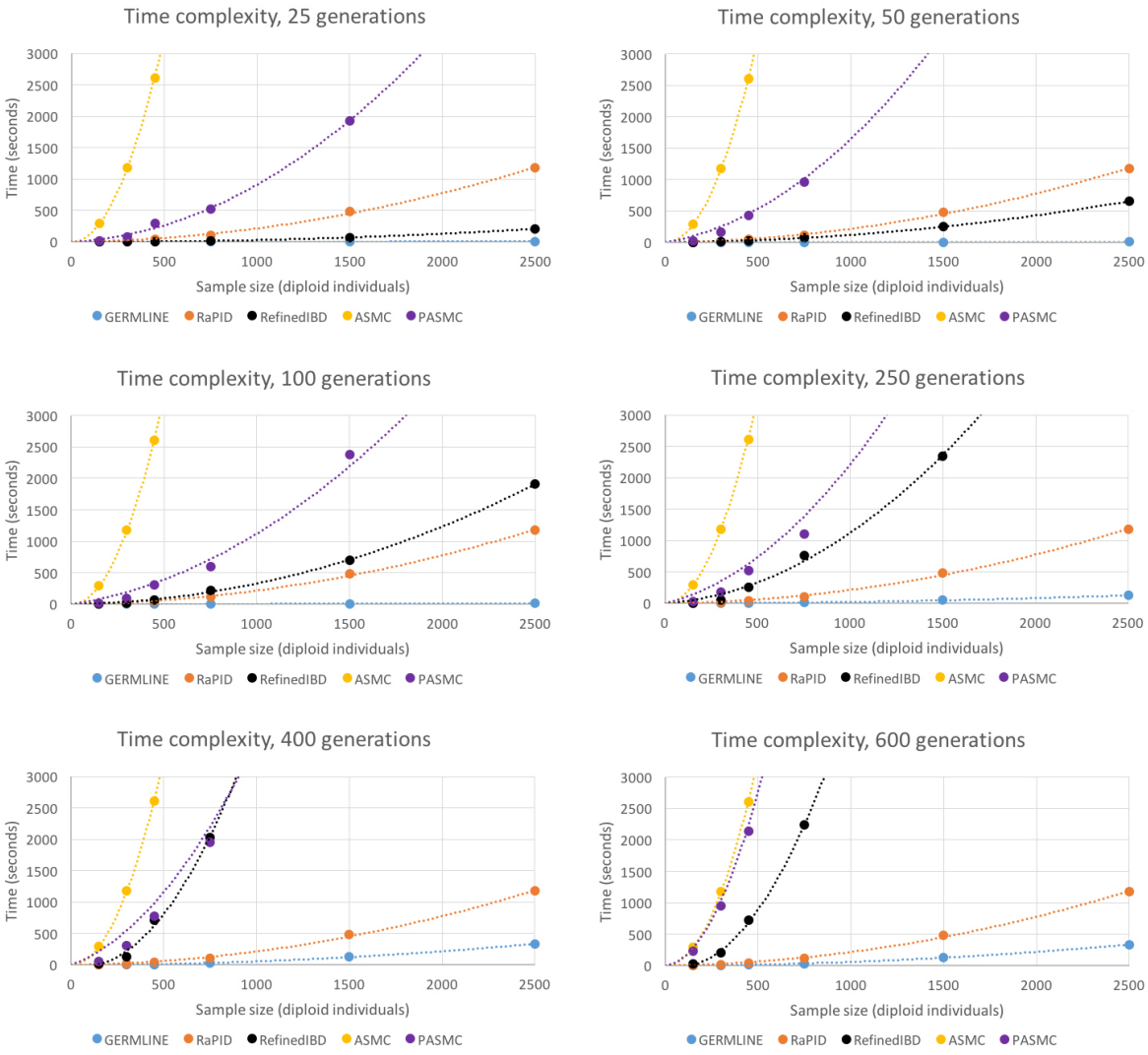


Figure 3.5: Time complexity for every IBD time threshold considered.

3.2.2 Time complexity

To compare the time complexity of all methods, we used the same six different datasets of 150, 300, 450, 750, 1500 and 2500 diploid individuals as before. The chromosome's length is still 30 Mb, with a European demographic history and recombination rates from a human chromosome 2. We compared the time complexity for each of the time thresholds t_{IBD} (25, 50, 100, 250, 400 and 600 generations). Note that only one thread was used for each method (no multi threading or any other parallelism techniques). See Figure 3.5. First, we can see that, as expected, the identification step makes ASMC faster. However, the further we look into the past, the closer PASMC's time complexity is to ASMC's. Moreover, PASMC doesn't scale as well as RefinedIBD for time thresholds close to present time and string-matching techniques are much faster. PASMC doesn't seem to be scalable, at least with the set of parameters we chose.

3.2.3 Memory

Because we do multiple runs of the PBWT algorithm, we store in memory all potential IBD candidate segments and increase their number of successes at each run. After that, they are all decoded by the HMM, batch by batch. Depending on the set of parameters, the identification step is more or less permissive, and the number of segments stored in memory differs. We used Valgrind [23], a programming tool for profiling, to measure the space memory complexity of our program. Valgrind allows us to measure how much both heap and stack memory our program uses through Massif tool. We then used Massif Visualizer [39], a very nice tool allowing us to visualize memory consumption through time (but we only keep the maximum memory consumption). We present in Figure 3.6 the memory complexity of PASMC for all time thresholds considered in the previous chapter. We can see that the memory consumption increases quadratically and is higher for high time thresholds, as the identification step is more permissive to allow very short IBD segments. For a time threshold of 100 generations, 2500 diploid individuals and a chromosome containing only 30 Mb, PASMC already requires around 2.7 Gb of RAM, which means it is not scalable in term of memory consumption.

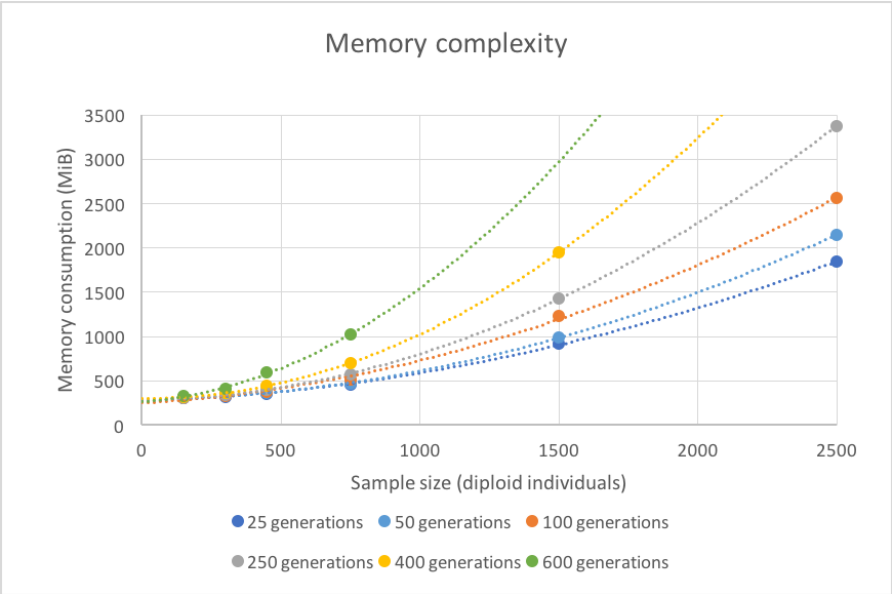


Figure 3.6: Memory complexity of PASMC algorithm.

3.3 Discussion

3.3.1 Downsides of PASMC

Not scalable The most important problem of PASMC is its inability to scale for large datasets, both in time and memory consumption, whereas that was the main motivation to implement it.

Too many parameters PASMC has probably too many parameters which makes it not user friendly. The goal is to get a program that can be used easily with as few parameters as possible.

Very strong correlation between parameters and time complexity The time complexity totally explodes and reaches ASMC's time complexity for some sets of parameters. Because the algorithm has a lot of parameters, the user may use a set of parameter with a very bad time complexity. This problem particularly occurs when the identification step is too permissive.

3.3.2 Future directions: a new and more promising approach

Our choice to base our identification step on RaPID's method was motivated by RaPID paper [22] which presents it as more accurate and faster than GERMLINE.

However, we tried various sets of parameters for RaPID and we never reached such results. RaPID was either less accurate or slower, and sometimes both. We will now move to GERMLINE's method as a preprocessing step instead of RaPID's method. First, the memory problem should disappear as there is no multiple runs in GERMLINE: as soon as an IBD segment is found, it would be directly decoded by ASMC. Secondly, in term of time complexity and accuracy, our results showed that GERMLINE was faster than RaPID and could reach a much better accuracy especially for recent common ancestors. In the next chapter, we present the results obtained using GERMLINE for the identification step.

Chapter 4

Germline - Ascertained Sequentially Markovian Coalescent

4.1 Method

We present here our new approach for IBD detection: the Germline Ascertained Sequentially Markovian Coalescent (GASMC) algorithm. As in the PASMC algorithm, there are two different steps in this method: an identification step and a validation step. The GERMLINE algorithm is first applied to identify IBD candidate segments, which are then filtered using the ASMC method. GASMC was built in C++.

Identification The identification step is done using GERMLINE algorithm (see Chapter 2). We did a grid-search over all possibles values of GERMLINE's parameters and decided to set the *bits* parameter to 64 and the minimum error parameter *err* to 0. The user can still use different values though. When an IBD segment breaks, instead of being outputted by GERMLINE, it is directly sent to ASMC for validation.

Validation The validation step is done using ASMC algorithm (see Chapter 2). ASMC builds an observation for each IBD candidate segment proposed by GERMLINE. Note that we are using here the same computational speed-ups and methodology as in the validation step of PASMC (see Chapter 3): ASMC adds 0.5 cM on both edges of all IBD candidates; the same definition of an IBD segment is adopted to increase the recall range; automatic vectorization is used while compiling the program; and ASMC decodes batches of observations. However, as there is no more multiple runs in the identification process, there is no need to merge segments here.

Grid search Please refer to GERMLINE's and ASMC's descriptions for parameters notations. In order to compare our method with the state-of-the-art algorithms in IBD detection, we did a grid search for each IBD time threshold t used in previous chapters. We provide here the best sets of parameters for each time threshold. Remember that the grid search was performed on an independent training dataset to both maximize the area under the precision-recall curve and to minimize the time complexity.

t	**25**	**50**	**100**	**250**	**400**	**600**
min_m	1.5	1	0.75	0.5	0.25	0.1
bits	64	64	64	64	64	64
err	0	0	0	0	0	0
t_{IBD}	25	50	100	250	400	600

4.2 Benchmarking

Accuracy We compared GASMC's accuracy with other algorithms using the percent auPRC improvement (see Definition 3.2.1) on the common recall range. We used the same 10 independent datasets we simulated and described in Section 3.4.1. Numbers in round brackets still represent standard errors on the 10 simulations. Note that if we decide to make the identification step very permissive (the extreme case would be when GERMLINE doesn't filter any pair of individual and any chromosomal region), we could reach the exact same accuracy as ASMC, but the time complexity would increase (similar to ASMC's time complexity in the worst case). Consequently, the maximum accuracy GASMC can reach is ASMC's accuracy. We decided here to use the set of parameters described in the previous paragraph, which is a trade-off between accuracy and time complexity. Results are presented in Figures 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6. In Figure 4.7, we randomly sampled precision-recall plots from the 10 independent datasets to illustrate the results. We can see that GASMC outperforms every existing method for recent IBD segments detection ($t < 100$ generations), except ASMC which appears to be its upper bound in term of accuracy. For higher time thresholds ($t > 250$ generations), GASMC suffers from its small recall range and performs a bit worse than RefinedIBD. We managed to increase the recall range by increasing the min_m parameter but the time complexity was too much affected. However, depending on the applications, we might prefer to be very strict and confident about the predicted IBD segments, which means a precision at least higher than 0.7, even if it means missing many IBD segments (low recall). We can see from the plots that GASMC's accuracy is very similar to RefinedIBD's for such precisions.

Moreover, note that 100 generations already corresponds to 3 5000 years (around 35 years per generation).

method	auPRC within recall range	percent auPRC improvement
GASMC	0.59 (0.02)	
GERMLINE	0.55 (0.02)	7.24 (0.86)
RaPID	0.46 (0.02)	31.50 (3.68)
RefinedIBD	0.53 (0.02)	12.13 (1.81)
ASMC	0.60 (0.02)	-1.80 (0.12)

Figure 4.1: Accuracy results obtained with an IBD time threshold of 25 generations on 10 independent datasets. The average common recall range is [0.07, 0.82].

method	auPRC within recall range	percent auPRC improvement
GASMC	0.56 (0.01)	
GERMLINE	0.51 (0.01)	10.73 (1.03)
RaPID	0.40 (0.01)	39.93 (3.54)
RefinedIBD	0.50 (0.01)	12.90 (1.46)
ASMC	0.57 (0.01)	-1.99 (0.09)

Figure 4.2: Accuracy results obtained with an IBD time threshold of 50 generations on 10 independent datasets. The average common recall range is [0.03, 0.71].

method	auPRC within recall range	percent auPRC improvement
GASMC	0.44 (0.01)	
GERMLINE	0.32 (0.01)	37.02 (2.76)
RaPID	0.28 (0.01)	60.91 (4.76)
RefinedIBD	0.41 (0.01)	9.34 (0.60)
ASMC	0.15 (0.01)	-2.16 (0.19)

Figure 4.3: Accuracy results obtained with an IBD time threshold of 100 generations on 10 independent datasets. The average common recall range is [0.01, 0.53].

method	auPRC within recall range	percent auPRC improvement
GASMC	0.26 (0.00)	
GERMLINE	0.04 (0.00)	504.37 (33.84)
RaPID	0.14 (0.01)	91.57 (8.67)
RefinedIBD	0.26 (0.00)	1.77 (0.31)
ASMC	0.27 (0.00)	-2.28 (0.18)

Figure 4.4: Accuracy results obtained with an IBD time threshold of 250 generations on 10 independent datasets. The average common recall range is [0.00, 0.29].

method	auPRC within recall range	percent auPRC improvement
GASMC	0.21 (0.00)	
GERMLINE	0.02 (0.00)	1196.13 (63.38)
RaPID	0.10 (0.01)	114.72 (11.19)
RefinedIBD	0.22 (0.00)	-4.45 (0.33)
ASMC	0.22 (0.00)	-4.86 (0.30)

Figure 4.5: Accuracy results obtained with an IBD time threshold of 400 generations on 10 independent datasets. The average common recall range is [0.00, 0.25].

method	auPRC within recall range	percent auPRC improvement
GASMC	0.16 (0.00)	
GERMLINE	0.02 (0.00)	806.90 (19.06)
RaPID	0.08 (0.00)	110.13 (9.86)
RefinedIBD	0.17 (0.00)	-6.29 (0.74)
ASMC	0.17 (0.00)	-6.60 (0.50)

Figure 4.6: Accuracy results obtained with an IBD time threshold of 600 generations on 10 independent datasets. The average common recall range is [0.00, 0.21].

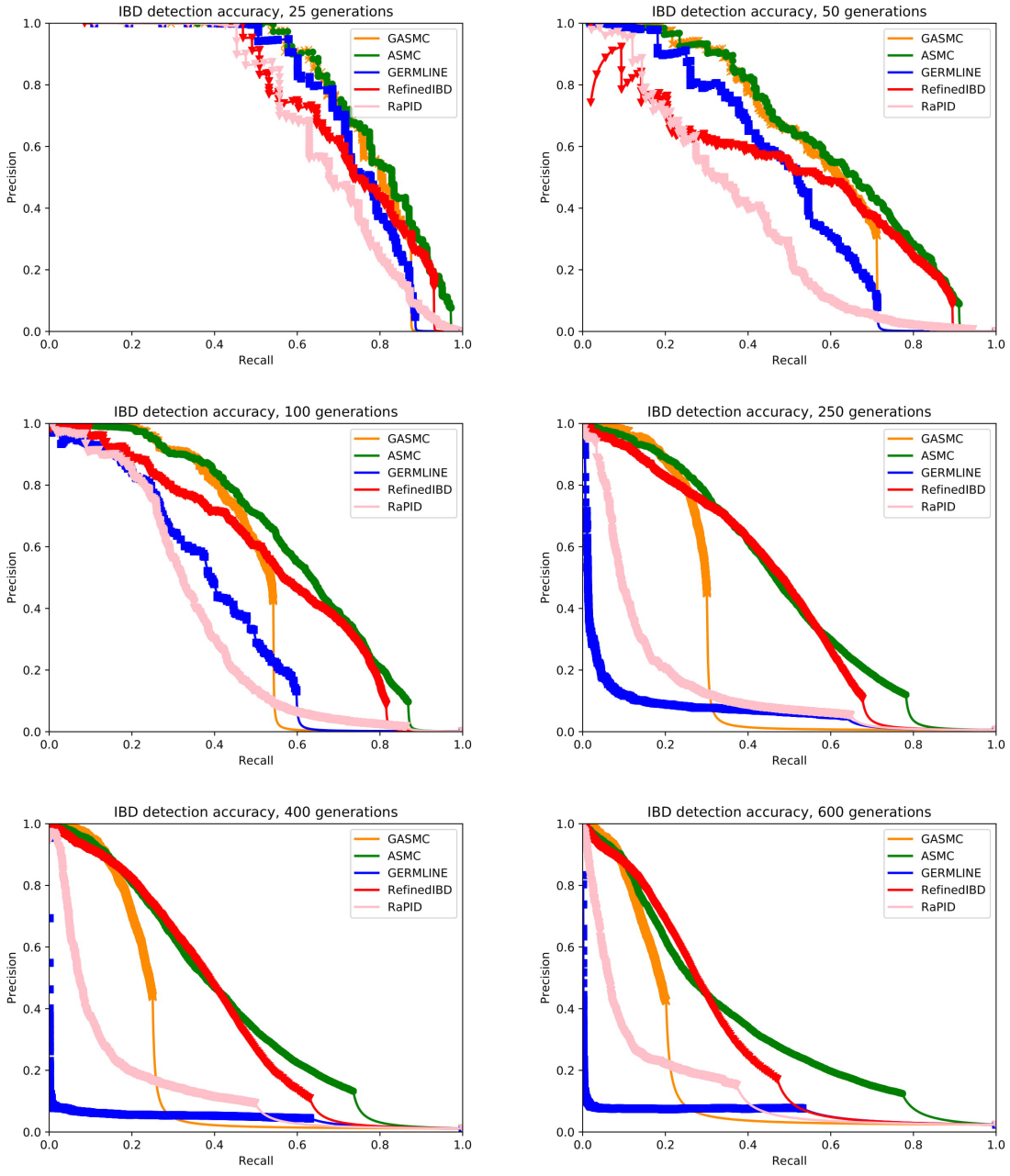


Figure 4.7: Precision-recall curves randomly sampled from 10 independent simulated datasets.

Time complexity Once again, we used the same six different datasets of 150, 300, 450, 750, 1500 and 2500 diploid individuals (see description of datasets in Chapter 2) to measure time complexity of each method. We compared the time complexity for each of the time thresholds t_{IBD} (25, 50, 100, 250, 400 and 600 generations) with the corresponding set of parameters found by the grid-search. As before, to get a fair comparison between every method, only one thread was used (no multi threading or any other parallelism techniques). Results are presented in Figure 4.8. As expected, the identification step makes ASMC faster. GASMC scales better than RefinedIBD no matter the time threshold considered and is even faster than RaPID for time thresholds close to present time (below 50 generations), where GASMC is almost linear in time. For higher time thresholds, we increased the minimum length parameter (see results from the grid search) to make the identification step more permissive, which explains why GASMC is slower.

Memory The results from the memory benchmark are presented in Figure 4.9. For details about how these results were computed, please refer to Chapter 3. We can clearly see that GASMC perfectly scales in term of memory. For example, for a time threshold of 100 generations, 2500 diploid individuals and a chromosome containing 30 Mb, GASMC only requires around 0.57 Gb of RAM, compared to 2.7 for PASMC. This huge difference is explained by the fact that we now need only one run of the identification step and we never store in memory the IBD candidate segments. Each IBD candidate segment is directly decoded by ASMC without being stored. Note that in the current version of GASMC, the data is loaded twice (in both the identification and the validation steps); this will be fixed very soon in the future version.

4.3 Discussion

GASMC results are very promising: it outperforms state-of-the-art methods in term of accuracy and scales very well in term of memory and time complexity. This new approach allowed us to solve the issues faced by PASMC. Future directions include working on a multi-threading version of GASMC, a binary output and applying GASMC on very large dataset such as the UK Biobank dataset to infer genetic relatedness on real individuals. According to our simulations, on a server with 200 cores, running GASMC on the UK Biobank to infer IBD segments for which the common ancestors lived between 100 generations ago and present time would require around 33 days,

with a precision around 0.8 and a recall around 0.4, while it would require around 500 days to run RefinedIBD with a precision around 0.7 and a recall around 0.4.

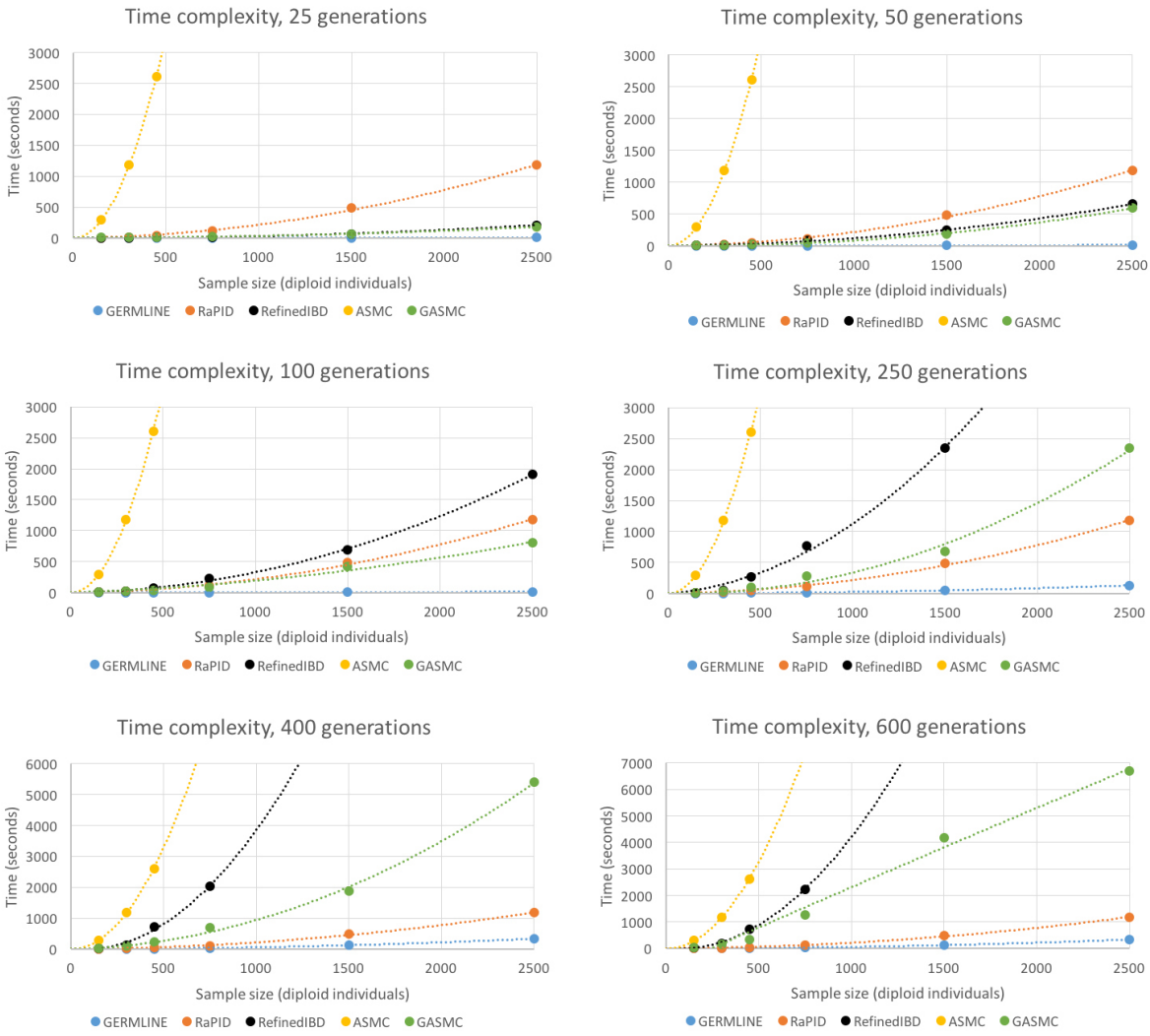


Figure 4.8: Time complexity for every IBD time threshold considered.

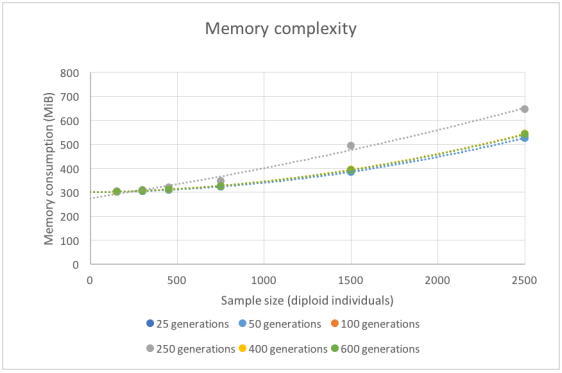


Figure 4.9: Memory complexity of GASMC algorithm.

Chapter 5

Conclusions

In this thesis we presented the basics of coalescent theory to introduce Identical-By-Descent segments in Chapter 1 and then provide in Chapter 2 an overview of state-of-the-art methods for detecting IBD segments (RaPID, GERMLINE, ASMC and RefinedIBD). We compared the accuracy using the precision-recall curve and the time complexity of the most recent and used algorithms. By doing that, we illustrated the accuracy-scalability trade-off in IBD detection. While some methods such as GERMLINE and RaPID scale very well to very large datasets and favor computational speed over accuracy, other algorithms such as RefinedIBD and ASMC achieve impressive results but don't scale as well. In order to overcome this issue, in Chapter 3, we decided to build a new method called PASMC aiming to bridge the gap between accuracy and scalability. We decided to add an identification step to the ASMC algorithm which outperformed every other method in term of accuracy in our analysis. To do that, we implemented the PBWT algorithm and tried to reproduce the RaPID algorithm as a preprocessing step. However, PASMC was not scalable in term of memory consumption and was not able to scale in time to large datasets. In Chapter 4, we tried a different approach and used GERMLINE algorithm as an identification step to build the GASMC algorithm. Our results were very promising: GASMC was more accurate than RefinedIBD, GERMLINE and RaPID, scales well to large datasets and doesn't face any memory consumption problem. The trade-off between scalability and accuracy has hindered IBD detection research as analysis on large datasets would only be possible by greatly lowering accuracy. Our proposed algorithm reaches state-of-the-art speed while being uncompromising on accuracy, therefore enabling large-scale IBD detection at the fraction of the most accurate algorithm run time, and with a similar accuracy.

Appendix A

IBD segments simulator - source code

We present here the source code of the IBD segments simulator introduced in Chapter 2. The simulator was implemented in Python on a Jupyter notebook.

IBD segments simulator

September 27, 2018

Juba NAIT SAADA

Assumptions :

- constant population size
- constant mutation rate
- constant recombination rate

```
In [99]: import numpy
         import matplotlib.pyplot as plt
```

```
In [100]: N = 10000 # 10000 diploid individuals, 20000 haplotypes
          lenTot = 50 # total lenght in cM of the chromosome
          mu = 10**(-8) # constant mutation rate
          r = 1.3*10**(-8) # constant recombination rate

          def simulator(genomeSize, popSize):
              position = 0
              recombTime = 0
              results = [['Length (centimorgan)','Coalescent time (continuous)',
                          'Recombination time (continuous)',
                          'Mutated sites (centimorgan)']]

              while(position<genomeSize):
                  coalTime = numpy.random.exponential(1) + recombTime
                  recombTime = numpy.random.uniform(0, coalTime)
                  length = numpy.random.exponential(1/coalTime)
                  mutationNb = numpy.random.poisson((length*2*coalTime*mu/r))
                  mutatedSites = []
                  for m in range(mutationNb):
                      mutatedSites.append(numpy.random.uniform(position, length))

                  position += length
                  results.append([position,coalTime,recombTime,mutatedSites])

              return results[:-1]

          results = simulator(lenTot,N) # running the simulation
```

```
In [101]: def extractFeature(x,indexFeature,typeFeature):
              if typeFeature == "mutation" :
                  feature = [len(x[0][indexFeature])]
              else :
                  feature = [x[0][indexFeature]]
              for y in range(1,len(x[2:])):
                  if typeFeature == "time" :
                      feature.append(x[y][indexFeature]*2*N)
                  elif typeFeature == "length" :
                      feature.append(x[y][indexFeature]-x[y-1][indexFeature])
                  elif typeFeature == "mutation" :
                      feature.append(len(x[y][indexFeature]))
              return feature

In [102]: lengthIBD = extractFeature(results[1:],0,"length")
          coalescentTime = extractFeature(results[1:],1,"time")
          recombinationTime = extractFeature(results[1:],2,"time")
          mutationsNb = extractFeature(results[1:],3,"mutation")

          print('--------------------------------')
          print('Number of IBD segments :', len(results))
          print('Average length of IBD segments (in cM) : ',
                numpy.mean(lengthIBD))
          print("Standard deviation of IBD segments' length (in cM) : ",
                numpy.std(lengthIBD))

          print('--------------------------------')
          print('Average coalescent time (in generation) : ',
                int(numpy.mean(coalescentTime)))
          print('Standard deviation of coalescent time (in generation) : ',
                numpy.std(coalescentTime))

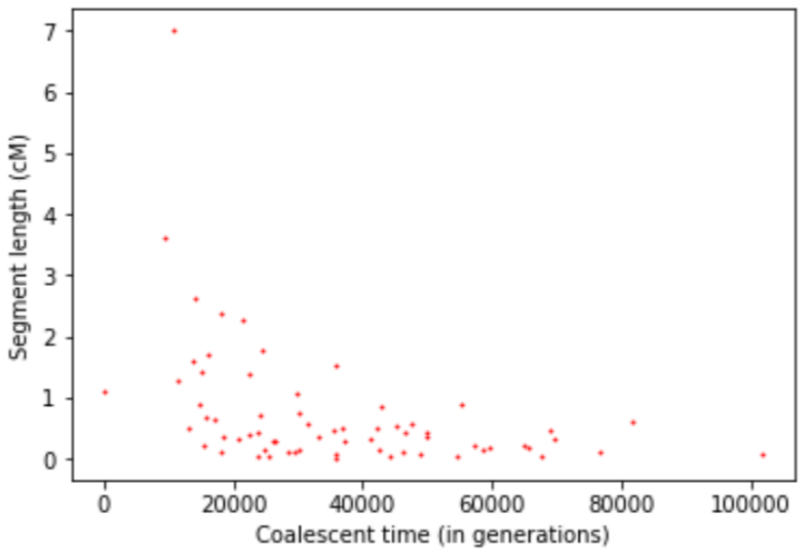
          print('--------------------------------')
          print('Average recombination time (in generation) : ',
                int(numpy.mean(recombinationTime)))
          print('Standard deviation of recombination time (in generation) : ',
                numpy.std(recombinationTime))

          print('--------------------------------')
          print('Total number of mutations on the chromosome : ',
                len(mutationsNb))
          print('Average number of mutations per IBD segment : ',
                numpy.mean(mutationsNb))

--------------------------------
Number of IBD segments : 69
Average length of IBD segments (in cM) :  0.728494562201
Standard deviation of IBD segments' length (in cM) :  1.04730551416
```

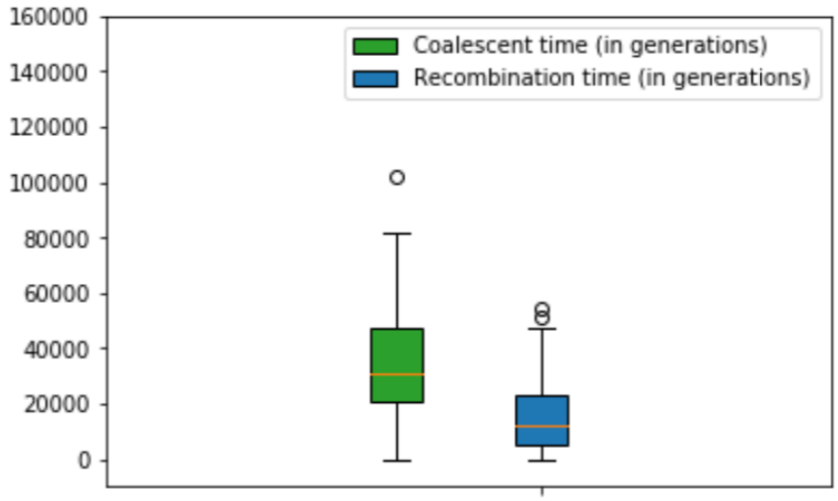
```
------------------------------
Average coalescent time (in generation) :  35850
Standard deviation of coalescent time (in generation) :  20095.2517296
------------------------------
Average recombination time (in generation) :  16306
Standard deviation of recombination time (in generation) :  12857.0526296
------------------------------
Total number of mutations on the chromosome :  66
Average number of mutations per IBD segment :  1.54545454545
```

```
In [103]: plt.plot(coalescentTime,lengthIBD,
                   'ro', markersize=1)
          plt.xlabel('Coalescent time (in generations)')
          plt.ylabel('Segment length (cM)')
          plt.show()
```



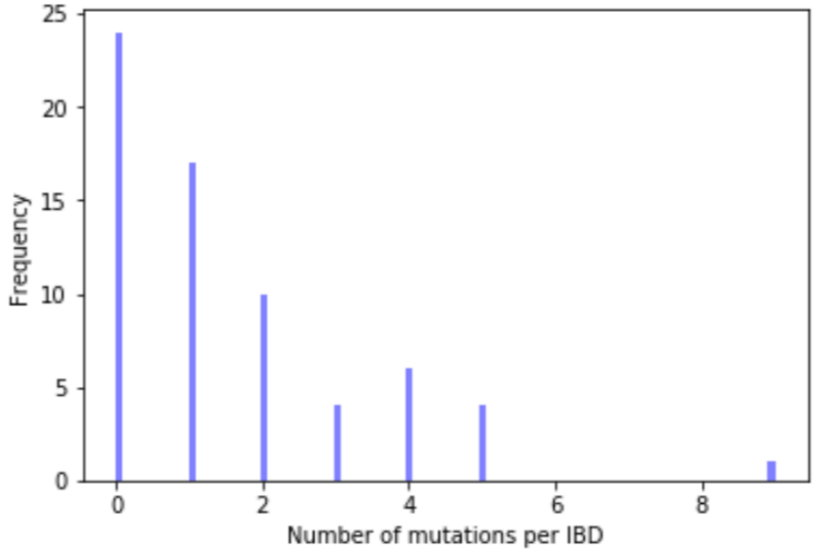

```
In [104]: fig, ax = plt.subplots()
          plt.figure(figsize=(10,10))
          bp1 = ax.boxplot(extractFeature(results[1:],1,"time"), positions=[2],
                           widths=0.35, patch_artist=True,
                           boxprops=dict(facecolor="C2"))
          bp2 = ax.boxplot(extractFeature(results[1:],2,"time"), positions=[3],
                           widths=0.35, patch_artist=True)
```

```
    ax.legend([bp1["boxes"][0], bp2["boxes"][0]],
              ['Coalescent time (in generations)',
               'Recombination time (in generations)'],
              loc='upper right')
    ax.set_xticklabels([''])
    ax.set_xlim(0,5)
    ax.set_ylim(-10000,160000)
    plt.show()
```




```
<matplotlib.figure.Figure at 0x11fb9f160>
```

```
In [105]: plt.hist(mutationsNb, 100, facecolor='blue', alpha=0.5)
          plt.xlabel('Number of mutations per IBD')
          plt.ylabel('Frequency')
          plt.show()
```


25
20
15
10
5
0
Frequency
0
2
4
6
8
Number of mutations per IBD

Appendix B

GASMC Manual

B.1 Installation

Extract GASMC and enter the extracted directory with the following command lines:

```
$ tar xzvf GASMC-v-1.tar.gz
$ cd GASMC-v-1
```

Then, you need to compile GASMC with:

```
$ cd SRC && make && cp GASMC ..
```

B.2 Run and input

To run GASMC, enter the following command line :

```
$ ./GASMC [inputs] [options]
```

Here is the list of inputs you need to provide to GASMC:

--inFileRoot *[path]* prefix of input files. Input files include a HAP file, a SAMPLE file, a MAP file and a PED file.

--outFileRoot *[path]* output file for maximum-a-posteriori distribution over pairs of individuals

--decodingQuantFile *[file]* decoding quantities file. A script (get_sfs.py) is provided in the extracted directory to build the CSFS file, which can then be used as an input of the PrepareCoalescentDecoding.jar program to build the decoding quantities file.

--timeThreshold *[positive integer]* time threshold in generations for IBD detection.

--minLength *[positive number]* minimum length in cM of IBD segments.

B.3 Output

GASMC generates a .ibd file in the specified location. Each line corresponds to a pairwise shared segment, with the following fields:

1. First individual identifier
2. First individual haplotype index (1 or 2)
3. Second individual identifier
4. Second individual identifier (1 or 2)
5. Chromosome number
6. Starting position of the IBD segment (inclusive)
7. Ending position of the IBD segment (inclusive)
8. Average maximum-a-posteriori over the whole segment

B.4 Options

Here is the list of options GASMC provides to the user:

--mode *[string]* (default=array) Decoding mode. Choose from {sequence, array}, depending on your dataset.

--bits *[positive integer]* (default=64) size of each slice (in markers) used for exact matching seeds in the identification step.

--errHom *[integer]* (default=0) maximum number of mismatching markers for a slice to still be considered part of a match in the identification step.

--batchSize *[positive integer]* (default=16) size of each batch for decoding.

Bibliography

[1] Katherine Brind'Amour and Benjamin Garcia. Wilhelm august oscar hertwig (1849-1922), available from http://embryo.asu.edu/handle/10776/1707. *Embryo Project Encyclopedia*, 2007.

[2] B L Browning and S R Browning. Improving the accuracy and efficiency of identity-by-descent detection in population data. *Genetics*, 194(2):459–71, 2013.

[3] S. R. Browning and E. A Thompson. Detecting rare variant associations by identity-by-descent mapping in case-control studies. *Genetics*, 190(4):1521–1531, 2012.

[4] Clare Bycroft, Colin Freeman, Desislava Petkova, et al. Genome-wide genetic data on $\sim$ 500,000 uk biobank participants. *bioRxiv*, 2017.

[5] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(5507):860–EP, 2001.

[6] The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[7] Ralf Dahm. Discovering dna: Friedrich miescher and the early years of nucleic acid research. *Human Genetics*, 122(6):565–581, 2008.

[8] Centre National de la Recherche Scientifique (CNRS). Les grands equipements scientifiques du cnrs; la competition prive/public : Celera genomics. [online]. *La Revue pour l'histoire du CNRS*, 2005.

[9] R Durbin. Efficient haplotype matching and storage using the positional burrows - wheeler transform (pbwt). *Bioinformatics*, 30(9):1266–1272, 2014.

[10] R.A Fisher. The genetical theory of natural selection. *Clarendon*, 1930.

[11] Alexander Gusev, Jennifer K. Lowe, Markus Stoffel, Mark J. Daly, David Altshuler, Jan L. Breslow, Jeffrey M. Friedman, and Pe'erl Itsik. Whole population, genome-wide mapping of hidden relatedness. *Genome Research*, 19(2):318–326, 2009.

[12] Jotun Hein, Mikkel H. Schierup, and Carsten Wiuf. *Gene genealogies, variation and evolution*. Oxford University Press, 2014.

[13] Maureen A. Hess, James G. Rhydderch, Larry L. LeClair, Raymond M. Buckley, Mitsuhiro Kawase, and Lorenz Hauser. Estimation of genotyping error rate from repeat genotyping, unintentional recaptures and known parentâĂŞoffspring comparisons in 16 microsatellite loci for brown rockfish (sebastes auriculatus). *Molecular Ecology Resources*, 12(6):1114–1123, 2012.

[14] Asger Hobolth and Jens Ledet Jensen. Markovian approximation to the finite loci coalescent with recombination along multiple sequences. *Theoretical Population Biology*, 98, 2014.

[15] Motoo Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):693, 1969.

[16] Heng Li and Richard Durbin. Inference of human population history from whole genome sequence of a single individual. *Nature*, 475(7357):493âĂŞ496, 2011.

[17] Heng Li, Bob Handsaker, Petr Danecek, Shane McCarty, John Marshall, et al. Bcftools, available from http://samtools.github.io/bcftools/.

[18] Giordano Liberi and Marco Foiani. The double life of holliday junctions. *Cell Research*, 20:611 EP–, 2010.

[19] Jonathan Marchini et al. A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78(3):437–450, 2006.

[20] Gilean A.T McVean and Niall J Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459):1387âĂŞ1393, 2005.

[21] Alistair Miles. Understanding pbwt, available from http://alimanfoo.github.io. 2016.

[22] Ardalan Naseri, Xiaoming Liu, Shaojie Zhang, and Degui Zhi. Ultra-fast identity by descent detection in biobank-scale cohorts using positional burrows-wheeler transform. *bioRxiv*, 2017.

[23] Nicholas Nethercote and Julian Seward. Valgrind: A framework for heavyweight dynamic binary instrumentation. *PLDI*, 2007.

[24] National Human Genome Research Institute NHGRI. Dna sequencing costs: Data. 2013.

[25] National Library of Medicine (US). Genetics home reference [internet], available from https://ghr.nlm.nih.gov/chromosome/1. *Bethesda (MD): The Library;*, Aug 2018 [reviewed Nov 2014].

[26] P. F. Palamara, T. Lencz, A. Darvasi, and I PeâĂŹer. Length distributions of identity by descent reveal fine-scale demographic history. *American Journal of Human Genetics*, 91(5):809–822, 2012.

[27] Pier Francesco Palamara. Population genetics of identity by descent. *Columbia University Academic Commons*, 2014.

[28] Pier Francesco Palamara. Argon: fast, whole-genome simulation of the discrete time wright-fisher process. *bioRxiv*, 2016.

[29] Pier Francesco Palamara, Jonathan Terhorst, Yun S. Song, and Alkes L Price. High-throughput inference of pairwise coalescence times identifies signals of selection and enriched disease heritability. *Nature Genetics*, 50(9):1311–1317, 2018.

[30] Francois Pompanon, Aurelie Bonin, Eva Bellemain, and Pierre Taberlet. Genotyping errors : causes, consequences and solutions. *Nature Reviews Genetics*, 6(11):847–59, 2005.

[31] Shaun Purcell and Christopher Chang. Plink, available from www.cog-genomics.org/plink/1.9/.

[32] Jared C Roach, Gustavo Glusman, Arian FA Smit, Chad D Huff, Robert Hubley, Paul T Shannon, Lee Rowen, Krishna P Pant, Nathan Goodman, Michael Bamshad, et al. Analysis of genetic inheritance in a family quartet by whole-genome sequencing. *Science*, 328(5978):636–639, 2010.

[33] Caitlin Smith. Dna microarrays: A trusted tool keeps evolving, availble from https://www.biocompare.com/editorial-articles/172195-dna-microarrays-a-trusted-tool-keeps-evolving. *Biocompare*, 2015.

[34] Jonathan Terhorst, John A Kamm, and Yun S Song. Robust and scalable inference of population history from hundreds of unphased whole genomes. *Nature Genetics*, 48:303 EP–, 2016.

[35] J. Craig Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001.

[36] J. Wakeley and P.R. Wilton. Coalescent and models of identity by descent. *In: Kliman, R.M. (ed.), Encyclopedia of Evolutionary Biology. Oxford: Academic Press*, 1:287–292, 2016.

[37] J. D. Watson and F. H. C Crick. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(6):737 EP–, 1953.

[38] Kenneth G Wilson. The molecular basis of crossing over, available from http://www.cas.miamioh.edu/ wilsonkg/gene07/chap12/holliday.html. *Genetic 342 A and B, Miami University.*

[39] Milian Wolff. Massif visualizer software. 2011.

[40] Sewall Wright. Evolution in mendelian populations. *Genetics*, 16(2):97, 1931.