UPPSALA
UNIVERSITET

UPTEC X 17005
Examensarbete 30 hp
Oktober 2017

# Haplotype Inference as a case of Maximum Satisfiability

## A strategy for identifying multi-individual inversion points in computational phasing

Ebba Bergman

UPPSALA
UNIVERSITET

**Teknisk- naturvetenskaplig fakultet**
**UTH-enheten**

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

# Abstract

## Haplotype Inference as a case of Maximum Satisfiability

*Ebba Bergman*

Phasing genotypes from sequence data is an important step between data gathering and downstream analysis in population genetics, disease studies, and multiple other fields. This determination of the sequences of markers corresponding to the individual chromosomes can be done on data where the markers are in low density across the chromosome, such as from single nucleotide polymorphism (SNP) microarrays, or on data with a higher local density of markers like in next generation sequencing (NGS). The sorted markers may then be used for many different analyses and data processing such as linkage analysis, or inference of missing genotypes in the process of imputation

cnF2freq is a haplotype phasing program that uses an uncommon approach allowing it to divide big groups of related individuals into smaller ones. It sets an initial haplotype phase and then iteratively changes it using estimations from Hidden Markov Models. If a marker is judged to have been placed in the wrong haplotype, a switch needs to be made so that it belongs to the correct phase. The objective of this project was to go from allowing only one individual within a group to be switched in an iteration to allowing multiple switches that are dependent on each other.

The result of this project is a theoretical solution for allowing multiple dependent switches in cnF2freq, and an implemented solution using the max-SAT solver toulbar2.

Handledare: Carl Nettelblad
Ämnesgranskare: Lisa Klasson
Examinator: Jan Andersson
ISSN: 1401-2138, UPTEC X 17005

# Populärvetenskaplig presentation

Många biologiska analyser grundar sig idag på data som behöver behandlas efter att det tagits fram på för att vissa analyser ska kunna genomföras, eller få bättre resultat. En sådan typ av data är genetiska sekvenser som tagits fram med metoder som genererar korta, men intressanta, sektioner av data. Dessa metoder ger resultatet för en position i en kromosom, och om det finns flera kromosomkopior närvarande med olika markörer i positionen så får en position flera möjliga alternativ. Resultatet för de flesta flercelliga organismer blir då att vi vet vilka två genetiska variationer som finns i en position, eftersom de har två uppsättningar kromosomer. Hälften ärvs från en förälder, hälften från den andra.

Ibland är det bara viktigt att veta vilka genetiska variationer som finns i en position. Till exempel inom sjukdomsgenetik så är blödarsjuka en sjukdom som bara kräver en förändring i en position. Andra sjukdomar är mer komplexa och det är kombinationen av flera olika genetiska variationer som ger upphov till symptom. Då är det viktigt att veta vilka variationer som har ärvts tillsammans och därmed samspelar för att orsaka sjukdomen. När man vet vilka sekvenser som kommer från samma kromosom ur ett kromosompar så säger man att man har hittat haplotyperna. Att korrekt sortera sekvenser till rätt haplotyper är viktigt för att slutsatserna som forskare drar från senare analyser som antar att haplotyperna är sanna ska vara så korrekta som möjligt. Haplotyperna är även viktiga inom andra forskningsområden, så som populationsstudier.

I dagsläget finns det många sätt att utföra den här sorteringen. Vissa bygger främst på statistik hos redan framtagna haplotyper, andra tar mer hänsyn till släktskap mellan de individer som undersöks.

Om vi har två familjer med barn, och med ett halvsyskon från pappan i familj 1 och mamman i familj 2 så kan vi utnyttja informationen från familj 1, familj 2 och halvsyskonet för att hitta haplotyperna. Säg att halvsyskonet i en position har två varianter av en gen, g1 och g2. Om alla i familj 1 bara har g1, men syskonen i familj två har både g1 och g2 så vet vi halv halvsyskonet fått g1 från familj 1 och g2 från familj 2. Vi vet också att mamman i familj 2 har g2 i den här positionen i en av sina kromosomer. Med liknande analyser och med hjälp av statistiska sannolikheter så kan vi hitta de flesta haplotyper. Eftersom det rör sig om stora mängder data som kräver den här typen av analyser så har många olika program utvecklats.

`cnF2freq` är ett sådant program som är unikt i att det kan behandla stora syskonskaror; vanligtvis så tar släktbaserade program hand om grupper om 2 eller 3, inkluderat föräldrarna. Det `cnF2freq` gör är att ha två sorteringar från början med samma värde. Sedan uppdaterar vikterna för hur troligt det är att haplotyperna är korrekta. Om det är väldigt osannolikt att haplotyperna för en position är korrekta så korrigeras detta genom att sekvenserna byter plats där. För att undvika att det sker konstiga växlingar, exempelvis att vi sätter halvsyskonet ovan till att ha fått g2 från familj 2 samtidigt som vi sätter halvsyskonets mamma i familj 2 till att inte ha g2, så kan bara en individ i en grupp ändras samtidigt. En begränsning blir då att om programmet har gjort ett felaktigt antagande som kräver att mer än en individ ändras samtidigt så kommer ändringen inte att genomföras.

Syftet med projektet har varit att ta fram en lösning för att kunna växla flera individer samtidigt, under förutsättningen att alla de indiver vi vill ska växla växlar. Lösningen ska sedan läggas in i det existerande programmet. Eftersom `cnF2freq` är mycket ovanligt i sitt sätt att angripa problemet jämfört med frekvent använda program som tar fram haplotyper så togs lösningen fram från grunden. Genom att göra en översättning av problemet till logiska regler har det anpassats till att kunna lösas med befintliga program, så kallade maximum-satisfiability-lösare.

Resultatet av projektet är en teoretisk lösning och en ny version av programmet som komplement till de tidigare versionerna. På så sätt har det här projektet utforskat en möjlighet att förbättra ett verktyg som kan utföra en typ av databehandling som är viktig för en mängd vetenskapliga undersökningar.

# Contents

# 1.Abbreviations

CNF conjunctive normal form
DNF disjunctive normal form
GWAS genome wide association studies
HMM hidden Markov models
IBD identity by descent
max-SAT maximum satisfiability problem
MDA multiple displacement amplification
PCR polymerase chain reaction
SAT satisfiability problem
SNP single nucleotide polymorphism

# 2.Introduction

Today there are many relatively affordable methods of generating information about genetic sequences, such as short read sequencing and Single Nucleotide Polymorphism (SNP) -chips (Gibson and Muse 2009). If different copies of a chromosome are present at the time of sequencing, there will be as many true suggestions for a position in the results as there were chromosomes to begin with. For most mammals this will result in a sequence of pairs rather than two separate sequences, since they possess two copies of each chromosome; one from each parent. If the sample to be sequenced only contains one set of chromosomes this is not an issue, as is the case with data from sperms or eggs in mammals. However, these samples may not be the desired or easily generated for a wide variety of reasons.

Hence while a lot of data can be generated relatively inexpensively, the data is not always a true representation of the genome. The alleles will be generated in unordered pairs, differently spaced out across the chromosome depending on which methods were used to generate the data, but always presented as an ordered row which will be called a sequence of markers in this report. From the generated sequence of markers the co-localisation of alleles corresponding to the separate chromosomes is not apparent, instead of two separate sequences for the two chromosomes there are several interpretations.

For the simple example of three positions we may have the pairs: AC, TG, AT. The possible pairs of sequences are then: {ATA, CGT}, {ATT, CGA }, {AGA, CTT}, {AGT, CTA}. The process of finding which pairs of sequences are most likely to be true is called phasing the genotypes or finding the haplotypes, and having the linkage between the alleles as they would appear in a chromosome is important to a wide variety of analyses. (Browning and Browning 2011, O'Connell et al. 2014, Snyder et al. 2015, Abecasis et al. 2002)

Two fields with good examples of this are disease studies and population genetics. For example when studying complex genetic diseases caused by several genetic variations from different parts of a chromosome knowing which variants are present in the same chromosome is an important step to finding the disease genes. (Browning and Browning 2011, Snyder et al. 2015) A population genetics linkage analysis, which derives from the frequency at which alleles are inherited together, naturally must begin with knowing which alleles are inherited together.(Nettelblad et al. 2009, Nettelblad 2011) Knowing the linkage or rather the linkage disequilibrium (LD) researchers can then move on to detect different selective pressures.

For genome wide association studies (GWAS) if the correct haplotype is known and a data point is missing, reference populations can help supplement the information gained in the experiment through the process of imputation (Nettelblad 2012a, O'Connell et al. 2014). Continuing on the above example: if the haplotypes are A*A and CGT, where * is a missing data point, we can refer to matches within a library of reference genomes to find that ATA is the most likely correct sequence and use this for the rest of the analyses.

To give a specific example of disease genetics and phasing genotypes *"Personal Omics Profiling Reveals Dynamic Molecular and Medical Phenotypes"* by Chen et.al (2012) is a complex study of how variations in health, and hopefully diseases such as cancers, can be detected. It was one of the first large studies into personalised medicine. Among other things, the study included whole genome sequencing for which the authors did computational haplotyping for the majority of the sequences with a well known program, Beagle and for the remainder used an in-house script. The details can be found in the extended experimental procedures accompanying the article. (Chen et al. 2012, Browning and Browning 2009)

Given that the haplotypes can be so important for research it is hardly surprising that several methods for phasing genotypes have been developed. There has been an increasing interest in experimental phasing, where the phases are part of the generated data (Snyder et al. 2015, Browning and Browning 2011). There are also plenty of computational programs available to predict haplotypes applying different approaches to the problem. Some take into account if the individuals studied are related, some do not. Some use Hidden Markov models (HMMs) for estimating the probabilities of certain haplotypes, some prefer to use heuristic rule based solvers. (Browning and Browning 2011, Nettelblad et al. 2009)

This project has focused on the program `cnF2freq` , which is able to analyse big groups of related individuals in an effective manner. Most haplotyping programs that make use of how individuals are related consider groups of two or three individuals. `cnF2freq` implements a way to divide a big family tree into overlapping smaller groups, called focus trees. These are analysed separately, but because they overlap information can propagate through the family tree in later steps. Therefore bigger groups than two or three can be handled efficiently and resolved accurately.

The phasing process in `cnF2freq` can be described as the program making a guess of what the haplotypes should be for each individual and then iteratively evaluating this guess, taking into account the individual's focus tree. If at some point the current guess is evaluated to be incorrect, the benefit of switching at a position in the haplotypes and thus making a new guess is evaluated. If it is beneficial, the haplotypes switch places, and so does every position downstream, see Figure 5.

The existing code evaluates the benefit of switching one individual assuming that no other individual switches, and the structure of the program makes sure that only one individual switches within the focus tree. (Nettelblad 2012b, 2011) If several individuals in the same group would benefit from a switch, the best suited one will switch first and the others will wait for the next iteration where new probabilities of switching are generated.(Nettelblad 2012b) In the initial development of `cnF2freq` there were no switches implemented, but when introduced the results improved drastically. However, when making a switch dependent on the fact that no other switches occur in the group the option of making several individuals switch at the same time is lost. Allowing this may again improve results drastically and has not been done before.

The program as is performs very well, but there is still room for improvement. When looking at the data my supervisor has found instances where a small part of the family tree has ended up with different haplotypes from the rest of the tree in such a way that it is not biologically credible. One theory is that small sections prevent each individual from switching so as not to become too different to its closest relatives. An interesting case to explore is if there are situations where it would be beneficial for several individuals to switch simultaneously if and only if the other individual switches. To allow these multiple dependent switches is the purpose of this project.

# 3.Background

## 3.1.Phasing

Haplotype phasing is the process of determining the parentage of an allele which allows us to do analyses where it is important to know which alleles are inherited together.

To put it into a different context, in many multi-cellular eukaryotes the *haploid* gametes are eggs and sperm created through *meiosis*. When these combine to form a zygote the cell is *diploid* with one chromosome set from each parent (Campbell and Reece 2010). A *haplotype* is then the sequence of markers or alleles that can be found on one chromosome (Gibson and Muse 2009). Haplotype phasing can thus be understood as the process of determining a marker's or allele's parental chromosome (O'Connell et al. 2014).

It should come as no surprise that this information is valuable in population genetics, and very useful for genome wide association studies (GWAS). For example, SNP-chips can be used to find out whether alleles are present in certain positions across the genome, and once the phases are known other analyses, such as linkage analysis, can gather information of whether the alleles are connected to diseases (Wang et al. 2005). There are several instances in disease studies where haplotype phasing is vital, as when a disease is caused by several genetic variations being inherited together or when there are epigenetic effects that are decided by which gender contributed with the gene copy. (Browning and Browning 2011, Snyder et al. 2015, Nettelblad 2012b)

Other areas that can benefit from haplotyped sequencing data is medical genetics and research into biological mechanisms. In addition to this haplotypes can be used for imputation; inferring and inserting missing markers given enough context and reference maps for the haplotype frequencies for the markers. This is commonly applied to GWAS data. (O'Connell et al. 2014, Browning and Browning 2011, Snyder et al. 2015)

To generate the haplotypes we can either isolate them directly in the lab or infer them using computational methods and programs. Whereas new experimental methods have started to grow in numbers in later years, computational methods have been used for a long time and have adapted to the changing landscape of sequencing and genetics. The cost of obtaining data with phased genotypes is today mainly affected by the cost of sequencing. It is more economical to use a less costly sequencing method and then apply computational phasing rather than to do experimental phasing by separating the phases in vitro before sequencing. (O'Connell et al. 2014, Snyder et al. 2015)

## 3.2.Experimental Phasing

There are several methods of doing experimental phasing. It is commonly known that computational phasing is less expensive, but experimental phasing comes with the benefit of lacking bias from previous sequences compared to frequency-based computational methods. It also has the unique ability to properly phase *de novo* alleles in individuals without descendants. (Snyder et al. 2015)

Experimental phasing tends to start with the physical isolation of a chromosome in order to only sequence one haplotype at a time. This can be done through dilution, sequencing gametes, or taking advantage of how chromatin interacts with chromosomes. Once a chromosome has been isolated, the next complex challenge is to amplify the material so that there is enough to be sequenced. The amplification techniques used vary, but well known techniques such as PCR (Polymerase Chain reaction) and MDA (Multiple Displacement Amplification) are fairly common.(Browning and Browning 2011, Snyder et al. 2015)

The methods can be further divided into sparse and dense approaches. Sparse approaches reach over great distances, but miss many heterozygous positions, while dense approaches accurately phase short blocks, but are unable to connect those haplotype blocks over greater distances such as a full chromosome. Dense sequences are useful when gene details are of interest, as is the case of clinical diagnosis of cystic fibrosis. Sparse techniques can be an advantage when it is the co-occurrence of multiple genetic variations that are of interest, such as in some forms of cancer. (Snyder et al. 2015)

## 3.3.Computational Phasing

If haplotypes are not available from experimental data, the genotypes can be phased computationally. The popularity of single nucleotide polymorphism (SNP) analysis and short reads sequencing has led to several computational methods being developed to estimate haplotypes, though computational haplotyping was important long before that.(Abecasis et al. 2002, O'Connell et al. 2014, Gibson and Muse 2009)

Early methods relied on using individuals with only one, or few, heterozygous markers for starting points and had limited coverage of the genome. As biological processes such as mutation and recombination were taken into account, models grew more complex.These statistical models usually use HMMs. (Browning and Browning 2011)

Computational phasing algorithms can be divided into whether or not they take the relatedness of individuals into account. In both cases there is a tendency to rely on certain haplotypes as starting points. For unrelated individuals the most common haplotypes are used, and for related individuals the shared haplotypes are used. (Browning and Browning 2011)

### 3.3.1.Related individuals

When investigating related individuals a common term used is Identity By Descent (IBD). This means that stretches of the genome will be identical between individuals because of their common ancestry and this information can be used when phasing the genotypes. (O'Connell et al. 2014)

Phasing related individuals can be useful for GWAS, population genetics, and finding disease alleles, possibly even complex diseases. (O'Connell et al. 2014) It is possible to ignore the relatedness, and use an algorithm for unrelated individuals. In this case Browning&Browning (2011) says that the phasing will be more accurate than for unrelated individuals using the same algorithm, but it may give results that are biologically improbable such as an abundance of recombination within a family.

Depending on how large the family groupings are within a population of interest different algorithms may be useful.(Browning and Browning 2011) For small groups of three BEAGLE and SHAPE-IT algorithms work well (Browning and Browning 2011) as does `cnF2freq` (Nettelblad 2011). What differentiates `cnF2freq` is that it easily handles large groups of siblings and uses information from the entire available family tree to correctly phase haplotypes.

### 3.3.2.Example study using haplotyping

To give a better understanding of how phasing genotypes can fit in a scientific inquiry I will give an example using a recent study about goats in Italy: ”The Valdostana goat: a genome-wide investigation of the distinctiveness of its selective sweep regions” by Talenti et.al (2017).

The purpose of the study was to find unique selective sweep regions in the breed Valdostana goat. Selective sweeps are when variation is eliminated or reduced in a genomic region due to selection strongly favouring an allele, thus making it more common in a population. As the allele gains frequency in the population, so will the region surrounding it on the chromosome. This will make a part of the chromosome appear uniform between individuals in the population, but with time recombination, mutation and other processes will break down the uniformity of the region. Regions of uniformity are therefore good indications of recent selective sweeps. (Browning and Browning 2011, Talenti et al. 2017)

The steps of finding the selective sweeps were in short:

1. Gather samples
   *From 15 different Italian breeds.*

2. Extract DNA from samples

3. Genotype
   *SNP-chip.*

4. Inhouse script for homozygotes at each locus for each pair possible
   *To find if any individuals were too closely related; none of the Valdostana goats were.*

5. Exclude some data
   *That do not meet standards set.*

6. Phase and impute data
   *Using Beagle.*

7. Analyse the data using different methods
   - Multidimensional scaling in two dimensions, in order to get clusters
   - Region of Homozygosity (ROH) comparison within and between breeds, long ROH is a strong indicator of a selective sweep
   - Fixation Index analysis, to asses population differentiation between breeds
   - Bayesian analysis, to find SNPs that explained variance
   - Haplotype based analysis, to find evidence of selection on SNPs

8. Interpret and discuss the results
   *Finding that several regions in the Valdostana breed have been selected for, most strongly a region on Chromosome 7 were immune system genes are found*

9. Draw a conclusion
   *Selective sweeps have been found, includes genes involved in the immune system development/ regulation. Further analysis is needed.*

Notice that in step 6 phasing is done. This prepares the material for analysis. Imputation, also done in step 6, helps to prepare the data by making it more complete, but relies on having phased genotypes in order to efficiently use reference genomes in order to fill in missing genotypes.

In step 7 haplotype based analysis naturally requires that the haplotypes are found. Another analysis, finding a region of homozygosity(ROH) across individuals, also requires phased genotypes. Here comparisons are made between chromosome sequences within the population, something that requiring knowing, and therefore finding, the haplotypes.

As mentioned above 15 breeds were sampled of which the Valdostana goat was the focus of the study. The data had to be generated, then processed, including computational haplotyping, and then run through different programs to be analysed. All details are of course available in the article itself. Haplotype phasing was done in this study in part so that missing data could be imputed, in part so that the data was in a format that was usable for later stage analyses. This is a common way of using phased genotypes in a scientific study.

## 3.4.Hidden Markov Models

A Hidden Markov Model (HMM) is a model of a stochastic process where the full content of the model is not explicitly available to the user at any one time, it is hidden. The idea is that a phenomenon which emits a signal can be approximated as a stochastic process. The process can in turn be described by a model that does the same thing, whether the process is tossing coins, predicting words, or phasing markers. It is a statistical model with parameters that can be determined, but where the signal is a random process relying on the parameters. A type of machine learning, a HMM's parameters can be trained with training data and then applied on a sample. For `cnF2freq` the parameters are trained with the Baum-Welch algorithm and continuously updated in iterations of `cnF2freq` as information propagates through the family tree. (Zvelebil and Baum 2008, Nettelblad 2012a, Baum et al. 1970, Jones and Pevzner 2004)

A HMM is built by states, with one being the current state. In each time-step the current state emits a signal and the process may move to another state.Which signal is emitted and which state becomes the current one is determined by using probability parameters. (Zvelebil and Baum 2008) In the context of `cnF2freq`, these states can represent the four possible combination of gametes, if it is assumed each parent is diploid and the child is the combination of one gamete from each parent. Instead of the process moving through a sequence generated in time, e.g. cointoss at time $t_i$, the HMM will be moving over markers in the chromosome. Traversing between states will therefore be

analogous with recombination. (Nettelblad 2012a) For example the individual is a product of two gametes one of which came from parent 1 so moving from a state with parent 1's first gamete to a stage with parent 1's second gamete will represent that recombination has taken place in parent 1.

The symbols emitted from the states will for `cnF2freq` be the observed sequence of markers. The emission probabilities will in this context be determined by two factors; sureness and skewness. Sureness is a parameter introduced to handle misinformation form the sequencing stage, or missing markers in general, and represents how certain it is that the emitted symbol is actually correct. Skewness on the other hand represents how strongly ordered the pair of markers observed is. If skewness is 0.5, the pair *xy* is not ordered, if it is 0 it is ordered *xy*, if it is 1 it is ordered *yx*, and any value in-between represents how likely it is to be ordered by how close it is to an extreme value; the closer skewness is to 0 or 1, the more certain the ordering of the markers is with 0.5 representing an unordered pair. In `cnF2freq` the Baum-Welch algorithm is used for training the skewness parameters. (Nettelblad 2012b,a, 2011, Nettelblad et al. 2009, Baum et al. 1970)

## 3.5.cnF2freq

`cnF2freq` is a program written by my supervisor Carl Nettelblad. `cnF2freq` is used for haplotyping related individuals and handles big sibling groups (Nettelblad 2012b). A basic overview of the program can be found in Figure 1. The reason big sibling groups are not a problem is that `cnF2freq` divides the data into focus trees. This is a way of breaking down the bigger problem of finding the correct haplotypes for each member of a family tree(Figure 2), to finding the best haplotypes with regards to the potentially much smaller focus trees (Figure 3,4).

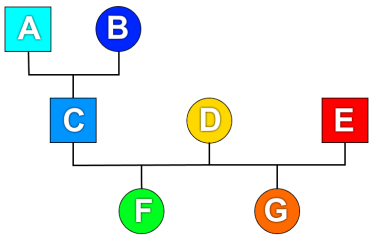


Figure 2: *A family tree of 7 individuals*.

All individuals that are related can be said to belong to a family tree, and a data set may contain several family trees. Each individual can be seen as the root of a focus tree that includes two generation of its direct ancestors. Siblings, uncles, and other individuals that are not direct ancestors will not belong to the focus tree, hence each focus tree will contain a maximum of seven members over three generations. There are going to be members of the family tree that belong to several focus trees that are not subsets of each other. That is, some individuals will create an overlap between focus trees. Thus, the focus trees will be tied together, and information can be propagated through the family tree. For example, two siblings will never be part of the same focus tree since we assume there is no incest, but the parents will belong to each sibling's focus tree and allow for information to be transmitted between the siblings via the parents.

At the start of phasing the genotypes `cnF2freq` sets the first heterozygous marker as each allele in the pair absolutely belonging to one parent. The program runs the possibility of the first allele belonging to parent 1 and 2 in two separate analyses until one option is deemed too unlikely to continue. In Figure 1, this would be equivalent to runing through the entire flow chart between *Genotypes* and *Haplotypes* in two separate instances. Each marker following the anchor is set to being unordered, represented by its skewness being 0.5. In iterations of the program a HMM gives a new estimate

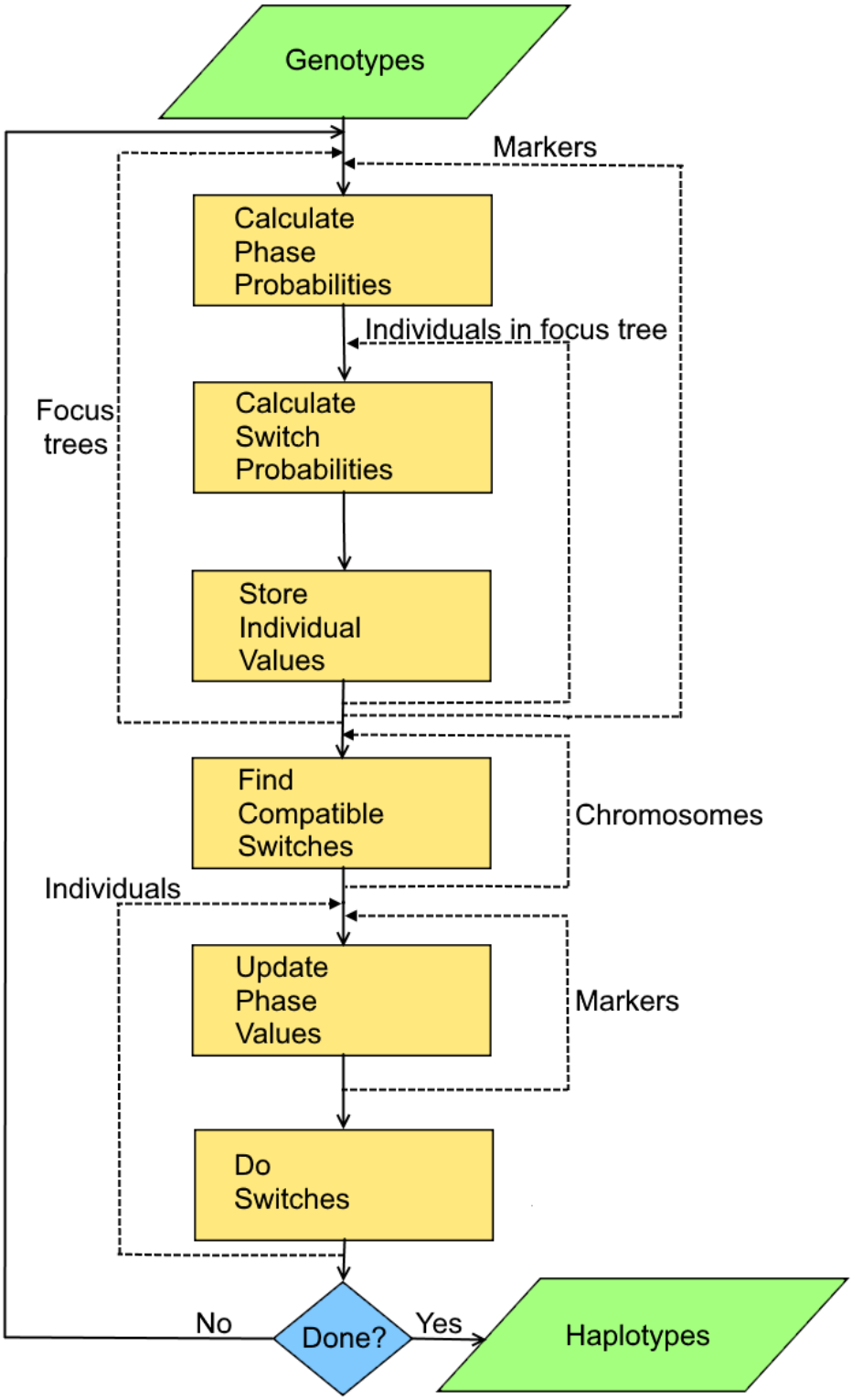


Figure 1: ***Flowchart for cnF2freq.*** *Solid arrows indicate the flow of the program, and dashed arrows iterative steps. The text close to the dotted lines show what is iterated over, for example "Update Phase Values" iterates over all markers. After each loop of the program there is a check to see if the Haplotype values have good enough probabilities that the program should exit, or if the loop has been done enough times that we would like to exit anyway. This is what the rhombus "Done?" signals. The data that is entered into the program is genotype data, and in the end haplotyped data is generated.*

for which phase each allele correctly belongs to, (*Calculate Phase Probabilities* in Figure 1), and the skewness is adjusted accordingly. The value goes towards 0 if the first allele in the pair belongs to the haplotype inherited from parent 1, and towards 1 if it belongs to parent 2's haplotype. The second allele in the pair will belong to whichever parent the first allele does not. (Nettelblad 2011, 2012a)

Phase probabilities are influenced by how the phases in the focus individual reflect the current predicted phases of the other individuals in the focus tree. The focus trees can all be run in parallel which lessens the run time of the algorithm. (Nettelblad 2011, Nettelblad et al. 2009)

### 3.5.1.Switches

As linkage decreases with distance, an allele far away from the anchor point may start to indicate that it actually belongs to the other haplotype. It then needs to be switched with the corresponding allele in the other phase. To switch phases the marker the allele belongs to, and every marker downstream of it, replaces its current skewness values $\gamma$ with $1-\gamma$, see Figure 5. For example a skewness value of 0.8 will change to 0.2, changing the phase for that marker's alleles. This is only done if this new sequence is seen as more probable over the whole pedigree (Nettelblad 2011). In this project this is referred to as a switch, but in other literature it may be referred to as an inversion.

It may happen that a skewness value is changed only in order to switch back to the same value it had before in the next iteration. In order to avoid oscillation a dampener is introduced. This method converges towards a result. (Nettelblad 2011)

It is possible that, as information propagates through a family tree, switching two parents and a child might lead to a better fit with the updated information. However, we may end up in a situation where if only the child changes it will be too different from the parental haplotypes, and if one parent changes it will be too different from its child, so the current code will not to switch any of them, as in Figure 6. This would not be a problem if we allowed multiple dependent switches, then all three could switch simultaneously.

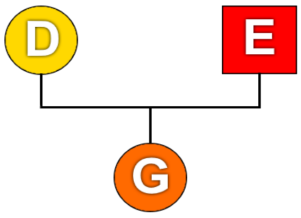


Figure 3: *The focus tree for individual G from the family tree in Figure 2. G has two parents, D and E, but no grand parents and so its focus tree will consist of three individuals. No siblings will be included in any focus tree, nor children.*

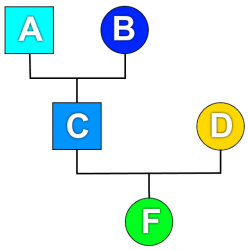


Figure 4: *The focus tree for individual F from the family tree in Figure 2. G has two parents, C and D, and two grandparents, A and B, so its focus tree will consist of five individuals.*

| s | | P1 | P2 |
|---|---|---|---|
| 0.0 | AB | A | B |
| 0.5 | CD | C | D |
| 0.5 | EF | E | F |
| 0.5 | GH | G | H |
| 0.5 | IK | I | K |
| 0.5 | LM | L | M |

The skewness s is neutral at the starting point. All pairs of alleles unordered, except the first Heterozygous one. Other markers are equally Likely to belong to either parent

| s | | P1 | P2 |
|---|---|---|---|
| 0.0 | AB | A | B |
| 0.1 | CD | C | D |
| 0.7 | EF | F | E |
| 0.6 | GH | H | G |
| 0.7 | IK | K | I |
| 0.3 | LM | L | M |

As skewness approaches 0 or 1 the marker pairs become more strongly ordered.

| Start | | Iteration 1 | | Iteration 2 | |
|---|---|---|---|---|---|
| s | | s | | s | |
| 0.0 | AB | 0.0 | AB | 0.0 | AB |
| 0.5 | CD | 0.1 | CD | 0.1 | CD |
| 0.5 | EF | 0.5 | EF | 0.5 | EF |
| 0.5 | GH | 0.6 | GH | 0.5 | GH |
| 0.5 | IK | 0.7 | IK | 0.8 | IK |
| 0.5 | LM | 0.3 | LM | 0.0 | LM |

If a shift improves probabilities most out of all possible within a focus tree, then put 1-skewness from inversion point onwards. After iteration 2, this will be between the two 0.5 values.

| Update values | Iteration 3 | Iteration 4 |
|---|---|---|
| 0.0 | 0.0 | 0.0 |
| 0.1 | 0.0 | 0.0 |
| 0.5 | 0.2 | 0.0 |
| 0.5 | 0.7 | 1.0 |
| 0.2 | 0.2 | 0.0 |
| 1.0 | 1.0 | 1.0 |

After iteration 4 the genotypes have been phased, resulting in the two sequences: ACFHKL and BDEGIM.

Figure 5: *Switching in one individual: How skewness values will order pairs, how switching will change the skewness and what the converging numbers represent.*

Currently `cnF2freq` only calculates switches on the basis that one individual will switch, and all the other individuals belonging to any focus tree the first individual belongs to stays the same. This is enforced by making sure only one individual in a focus tree switches in an iteration. (Nettelblad 2012b).

## 3.6.Multiple Dependent Switches

The goal of this project is to allow any group of sizes between zero to all individuals to switch at the same time, evaluating the groups on the basis that any one not in the group is not allowed to switch thus moving the focus from finding the best individual to switch to finding the best group to switch. For an illustration of a situation the new program is able to solve, see Figure 7.

*Multiple dependent switches* will in this report represent the option of switching several individuals belonging to a focus tree in a single iteration. The likelihood change induced of a group of individuals switching will be calculated based on all of the individuals in the group switching and no other scenario. Therefore the probability cannot be extended to a subset of the group.

It is important to note that a multiple dependent switch is not simply a number of individual switches being combined. Instead it is a set of individuals that are evaluated together, switching if and only if all the individuals in the set switches and no other individual in their focus tree switches.

## 3.7.Evaluating Methods for Computational Phasing

In order to compare how well any of the above methods perform several different metrics can be used. Some of these rely on having gold standard data which can be derived from nuclear families or experimental phasing. (Browning and Browning 2011). A method that does not require data that has external validation, such as gold standard data, is cross validation. This process masks known markers and then lets the program impute them (Browning and Browning 2011). A measurement of how often the correct marker is imputed is therefore independent of external data. This would be an interesting method to use if the code is tested on larger data sets. Other methods, such as haplotype accuracy and switch error, are feasible, but rely on gold standard data being available. (Browning and Browning 2011)

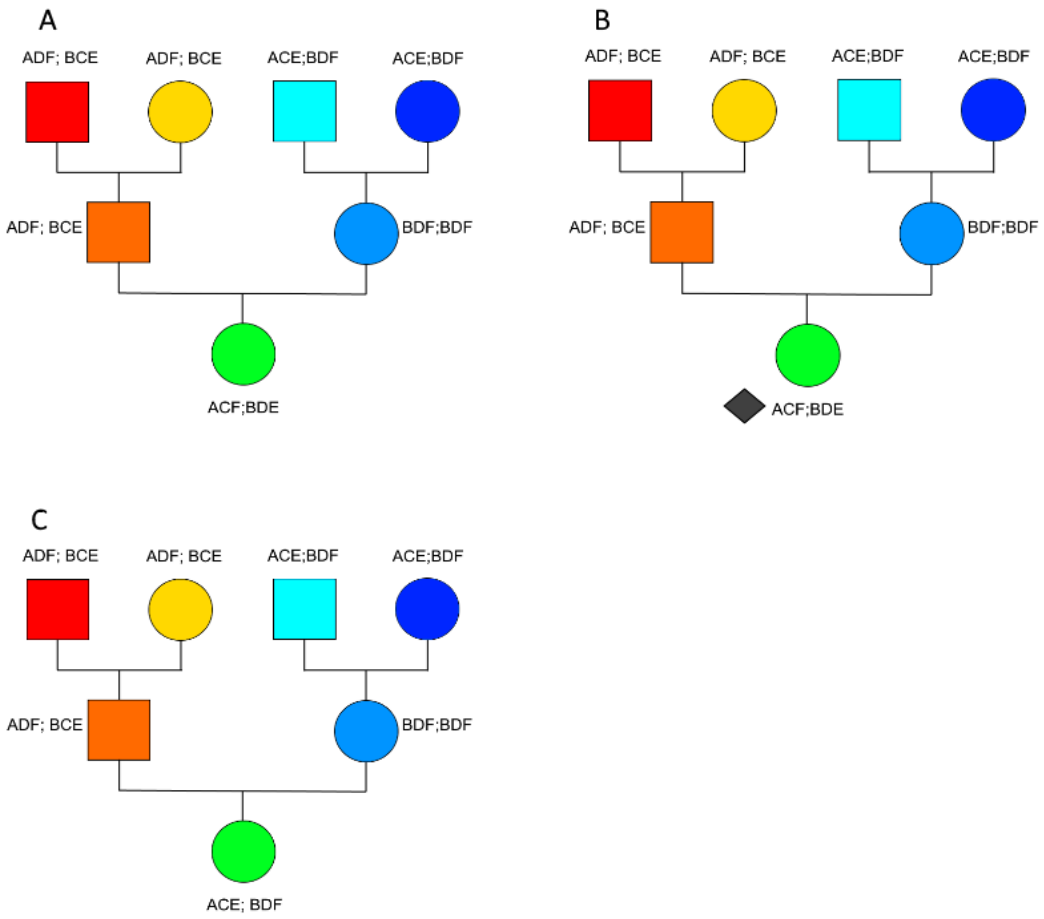


Figure 6: **Individual switches not resolving the phases of a biologically improbable family tree** *An illustration of a 7 member family tree of 3 generations with a 3 marker sequence being phased.* Step A: *The tree is not correctly phased. The green circle does not have a haplotype that corresponds to either of its parents.* Step B: *A switching point is determined in the green circle, represented by a grey rhombus.* Step C: *No beneficial switching points can be found. The green circle still does not have a haplotype that corresponds to the left hand side of its family tree, the orange side. All three individuals in the orange side would need to switch in their second position in order for the haplotypes to be consistent over the family tree. It is unlikely that the orange box would want to switch its haplotypes so that it no longer matches its two parents. Similarly the parents to the orange square will be locked by its child. Biologically improbable haplotypes will be result for parts of the family tree.*

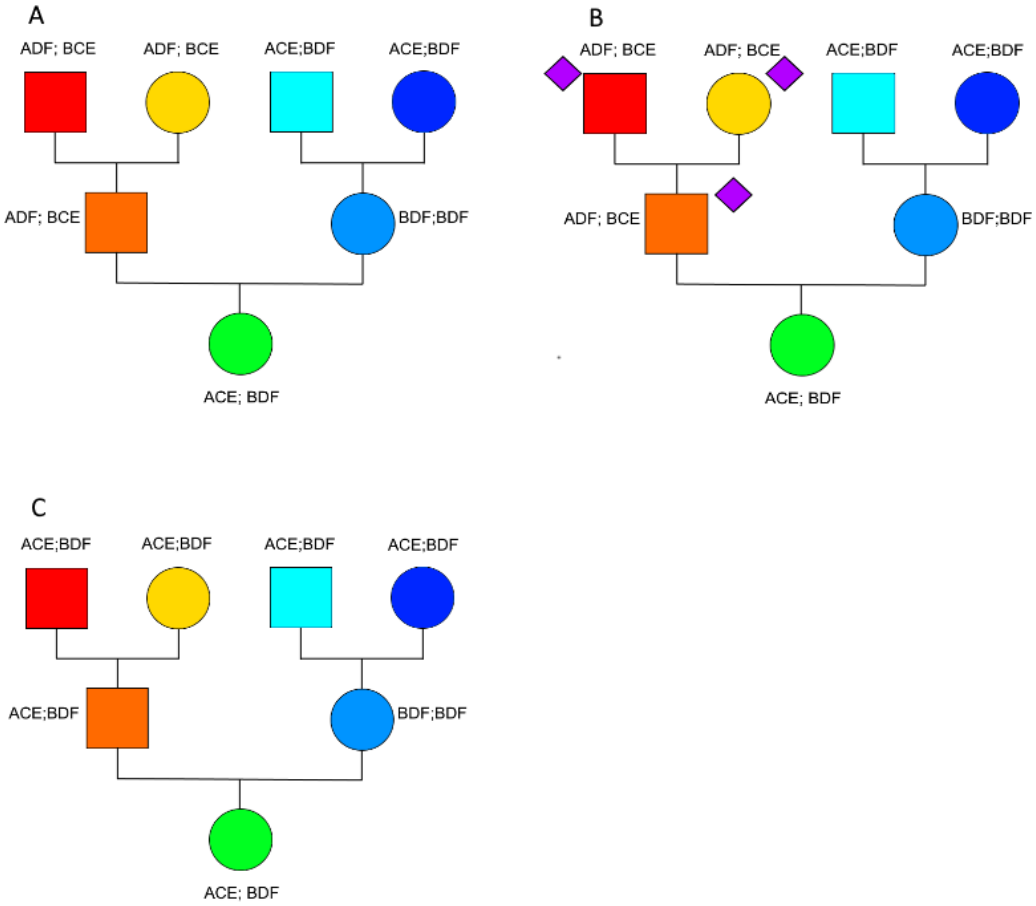


Figure 7: **Multiple dependent switches resolving the phases of a biologically improbable family tree** *An illustration of a 7 member family tree of 3 generations with a 3 marker sequence being phased, and a possible solution to the problem in Figure 6.* Step A: *The tree is not correctly phased. The green circle does not have a haplotype that corresponds to the orange left hand side of its family tree.* Step B: *The most beneficial switch has three switching points represented by purple rhombus. This is now possible since in addition to each individual switching independently multiple dependent switches are allowed.* Step C: *The genotypes have been phased in such a way that the whole family tree is resolved.*

# 4.Methods

Most of this section introduces the processes of switching in cnF2freq, the problems this system might pose, and the solution that was chosen from a more practical perspective. Last in this section there is a summation of what the first four weeks of this project was spent on, including some discussion of early models.

For the remainder of the report it is worth remembering some of the basic concepts of logic, as introduced in Table 1. This formalism is described in further detail in the *Theory* section, where I will also show the valid transition between different logical formulas that we need to make in order to have a functional solution.

Table 1: A summary of common and relevant logical concepts. These are some of the basic concepts in logic, for further reading see Barwise (2002) *Language and Logic*

| Logic | Explanation |
|---|---|
| 1, T | true |
| 0, F | false |
| $\wedge$ | and, conjunction. |
| $\vee$ | or, disjunction |
| $\neg$ | the negation of a variable, clause or formula |
| $\leftrightarrow$ | equivalence, is equivalent to |
| atomic sentence | a structure that cannot be broken down further, can be set to true or false |
| literal | is an atomic sentence or the negation of an atomic sentence |
| clause | a conjunction or disjunction of literals |

## 4.1.Details on switch implementation in cnF2freq

`cnF2freq` generates the probabilities needed in order to evaluate which individual in a focus tree to switch using a HMM (see Figure 1). The likelihood change induced by switching a marker in an individual is calculated under the assumption that no other switches will occur. As the HMM calculates this based on the current focus tree, no more than one individual that belongs to a focus tree can be switched in the same iteration. When all possible switches have been calculated, the non overlapping switches that most improve the solution are chosen. If an individual is part of a focus tree where another individual is switched, it cannot be switched in any focus tree at all for the duration of the iteration, as this would violate the assumption used to calculate the improvement the switch would bring.

Thus a switch will be made if and only if it is the best possible switch in all the focus trees it is a part of, and the switch makes the new haplotypes more likely than the current state. No two individuals that are part of the same focus tree can switch at the same time.

These constraints prohibit switches that are dependent on several individuals switching simultaneously. If we were to formulate this as a set of logical clauses, starting out from the very small case of a two individual focus tree like the one in Figure 8 we could state our options in the form found rows 1,2, and 3 in Table 2.

Table 2: How to phrase some options and their clauses for individual dependent switches for the focus tree in Figure 3

| Purpose | Logical Clause | Read |
|---|---|---|
| Do not switch in any individuals | $\neg$A$\wedge\neg B$ | ”no switch in A and no switch in B” |
| Switch only in individual A | A$\wedge\neg B$ | ”switch in A and no switch in B” |
| Switch only in individual B | $\neg$A$\wedge$ B | ”no switch in A and switch in B” |
| Switch both indvidual A and B | A$\wedge$B | ”switch in A and switch in B” |

The goal is to expand `cnF2freq` to allow for switches to occur in any number, from none to all individuals, in one iteration, while keeping the current functionality of the program.The problem is then divided into the following: how to construct rules that describe the possible changes, and how to choose which changes to make using these rules. The rules should be weighted in accordance to the score from the HMM that has already been implemented.

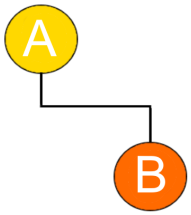


Figure 8: *A two individual focus tree for individuals A and B*

Starting with constructing the rules, an intuitive way is to use all the clauses described in the four rows of Table 2. We would then want to find weights for these clauses, and find which rules would be the best to fulfil. Or rather, which variables should be set to true, and their corresponding individuals switched. This leads us to the weighted Maximum Satisfiability solver.

## 4.2.Maximum satisfiability problem

The satisfiability problem (SAT-problem) consists of finding whether there is such an assignment of variables that a logical formula evaluates to true, i.e. if it is satisfiable. The logical formula in question must follow certain rules. Its clauses must be a disjunction of literals. A literal is a boolean variable or the negation of a boolean variable. The formula must consist of conjunctions of clauses. Eg. $\psi = C_1 \wedge C_2$ where $\psi$ is the formula, $C_1$ is the clause $l_1 \vee l_2$, $C_2$ is the clause $l_1 \vee -l_2$ and $l_1$, $l_2$ are literals of boolean variables. (Narodytska and Bacchus 2014, Xing and Zhang 2005, Larrosa et al. 2008)

If it is not possible to satisfy the entire formula we can attempt to maximize the number of clauses that can be satisfied by a variable assignment, and thus solve the Maximum Satisfiability problem (max-SAT)(Larrosa et al. 2008, Xing and Zhang 2005).This can be further extended to weighted max-SAT where each clause is assigned a weight representing the cost of not satisfying the clause. The sum of weights of falsified clauses is the cost of the variable assignment. The weighted max-SAT problem aims to minimize this cost (Larrosa et al. 2008).

Since the sum of the weights of all the clauses in the logical formula remains the same regardless of variable assignment, we can rephrase this to the weighted max-SAT problem being the problem of finding the maximum sum of weight from satisfied clauses (Xing and Zhang 2005). In weighted max-SAT an upper bound $T$ can be defined such that all clauses that has the weight $T$ must be satisfied. Such clauses are known as hard clauses, and soft clauses all have a weight $w < T$.(Larrosa et al. 2008)

In our case the variables are a representation of a marker in an individual. If the variable is true it indicates that a switch should occur in the individual in this marker. As we will never insist on a switch occurring all of our clauses will be soft. The optimal truth value assignment will be calculated for each marker over all individuals, and then the assignment with the lowest cost will represent the switch(es) to be made. It is possible that the optimal assignment is equivalent to no switch being made. The weights can be calculated using a slight adaptation of the weights that are already generated in the old version of cnF2freq.

The max-SAT problem is unlikely to have a solution that can be formulated as an algorithm which can be solved in polynomial time; it is NP-hard. The max-Sat problem builds on the boolean satisfiability (SAT) problem which is NP-hard for more than two literals (Xing and Zhang 2005). As it is NP-hard, building an efficient solver is a project in itself. Fortunately the max-SAT problem is not confined to any one field, and as such there is great interest in developing good solvers.(Xing and Zhang 2005, Larrosa et al. 2008, Narodytska and Bacchus 2014) Such solvers can be entered into the Evaluation of Max-SAT Solvers competition (http://www.maxsat.udl.cat/16/index.html) .

Some of the solvers are made available in an easy to use code format, and they all have to fit the competition's standard input and output formats.These standard input and output formats were the reason we decided that this project would focus on developing a solution that used one of the solvers using these formats. The hope is that a new solver will be easily fitted into the solution of this project if a new solver is picked, preferably after an evaluation of which solver fits `cnF2freq` problems best. Such an evaluation is beyond the scope of this project.

### 4.2.1. `toulbar2`

One solver that was entered into the competition in 2013 is `toulbar2`. `toulbar2` comes well documented and is easily installed on Ubuntu 16.10 as well as other linux versions. Due to the apparent ease of use it was chosen to be the solver used for this project.(Hurley et al. 2016)

It uses the standard input format of .wcnf which can be envisioned as:

```
c
c This is a comment
c
p wcnf nbvar nbclauses
```
$w_1\ l_1\ -l_2\ 0$
$w_2\ l_1\ -l_2 - l_3\ 0$
$w_3\ -l_3\ -l_2\ 0$
$w_4\ l_1\ -l_3\ 0$

Where $nbvar$ is the number of variables in the formula, $nbclauses$ is the number of clauses in the formula $w_i$ is a weight, and $l_i$ is a literal. Rows beginning with c are not processed and can be used to leave comments. The zeros at the end of each line is a requirement to correctly process the file, $p$ and $wcnf$ indicate that this file should be processed as a weighted max-SAT problem.

## 4.3.Alternative paths

The first part of this project was dedicated to modelling the problem. The purpose was to better understand the problem as well as finding a model that could then be used to derive a solution. The first idea was to model the switches or the focus trees as graphs and use some sort of search method to find the best possible switch. Quite some time was spent on learning, or re-learning, graph theory and starting to develop strategies for a solution. At the same time I also played around with a couple of rule based solutions.

One thing that I found difficult to work out in either case was how to do a model and solution that did not then create a further step of problem solving. For example, one basic idea was to use the individuals as nodes and weight edges between nodes to represent switching several individuals. Had switching multiple individuals at the same time been the same as doing several single individual switches it would have been possible to travel along the edges to find a path with a maximum weights, setting some limitations to the path. However, in `cnF2freq` the calculated benefit of doing a single switch is dependent upon no other individuals switching. So then there was a need to change that, and finding a way to tell the HMM to calculate the benefit of switching several individuals at the same time.

It was while focusing on how to express these switches to the HMM that I really focused on logical rules. Formulating ”I want to switch these two, but no one else” or ”I want to switch these three and no one else” in logical rules I viewed them as independent clauses written as *If (individuals that should switch) Then Not (all other individuals)*.

Another possibility in the graph would be to insert nodes for every tuple, or tripple, and use this as a starting point to see if there was a need to do multiple switches on this level. My supervisor, when looking at the data and deciding to explore multiple dependent switches, saw a pattern that indicated that small sections of the family tree got stuck with the wrong haplotypes, as exemplified in Figure 7, so this seemed like a reasonable start.

When I was introduced to max-SAT as a logical problem rather than from a graph-search perspective I started on what would become the solution. Now I found the easier way of expressing my clauses, namely as specifying all of the switches as individual literals joined with conjunctions and simply negating those who would not be switched. The next step to join them with disjunctions was very intuitive. Making the translation between conjunctive normal form and disjunctive normal form took a bit longer, especially since I was not familiar with the terms or their symmetries to start with. My supervisor’s instruction to think of them as sets made the translation a lot easier. Eventually I had the full theoretical solution and could start on the implementation right after my mid-project presentation.

# 5.Theory

The first focus of this project was to formulate a theoretical solution for making `cnF2freq` handle multiple dependent switches. An important step towards arriving at the full solution was to express the switches as logical clauses. Here I describe how this can be done, what we need in order to use an existing solver as presented in Methods and how to transition from the first logical clause set up to the second.

There are two option for each individual, to switch or not to switch. All possible combinations number $2^n$ for $n$ individuals. Since the focus trees contain at most seven individuals we will have no more than $2^7 = 128$ possibilities, and as a minimum there will be $2^1 = 2$ possibilities. For an easy overview of the solution I will use two individuals with $2^2 = 4$ switching possibilities. The extension to any other numbers is trivial.

## 5.1.Switches as Logical Clauses

When setting up logical expressions, it is good to keep in mind some common symbols and concepts, as seen in Methods. Our smallest building blocks will be atomic sentences which cannot be broken down further and can be set to either true or false (Barwise and Etchemendy 2002) . In this case a switch being made in an individual is an atomic sentence. To better represent whether a switch will occur or not we will use a literal, which is an atomic sentence or its negation (Barwise and Etchemendy 2002) . For example if literal A is true the atomic sentence A is true so a switch is made, but if the literal ¬A evaluates to true the atomic sentence A is false and so a switch is not made.

Just as in regular arithmetic, logic has a priority list of evaluations or order of precedence. In order: parenthesis, negations, conjunctions, disjunctions. Parentheses work as in mathematics; evaluation of their content has precedence over other evaluations. For both priority and scope it may be useful to think of negations as corresponding to having a number be negative, conjunction to multiplication and disjunction to addition. For clarity parenthesis will be used to separate disjunctions and conjunctions in all following examples.

To give an example, take two individuals and assign them the literals A and B. If we only want to switch A that is expressed as the clause: $A \wedge \neg B$. We want to switch in A and *not* in B. As all conjunctions this holds true if and only if we have a specific value assignment, in this case if A is true and B is false.

One easy way of proving this is using a truth table. A truth table will consist of columns, one for each atomic sentence as well as one for each formula. Each row will represent an unique combination of truth values for the atomic sentences. As previously discussed this specific case has four possible combinations, and therefore the truth table will have four rows. In this report each literal, operator, and clause will have a column of truth values.The column that represents what the largest scope evaluates to will be red, to make it easier to follow along.

Table 3: The truth table of the clause A ∧ ¬ B with the clause's values in red

| A | B | ( A | ∧ | ¬ | B ) |
|---|---|---|---|---|---|
| T | T | T | F | F | T |
| T | F | T | T | T | F |
| F | T | F | F | F | T |
| F | F | F | F | T | F |

In Table 3 the first two columns represent the truth assignments of the atomic sentences A and B. Looking at the first row, the first two columns set up that both A and B are true. These values are then copied to be under the atomic sentences in the clause A ∧ ¬ B. Next ¬ B will evaluate to the opposite of B, in this case false, so a 0 is put beneath ¬ in this row. Finally the entire clause is true if and only if both sides of ∧ , within the scope, evaluates to true. Since ¬ B is false, so is the clause. Looking at the column beneath ∧ it is easy to identify that the clause is only true for the second row where the atomic sentences A and B evaluate to true and false respectively.

The more visual minded reader may find it easier to view figures such as Figure 9 which will be referred to as truth squares. These are possible since we will only use examples with two atomic sentences and thus only two dimensions are needed to illustrate every possible combination of truth values. Each side will be divided into two, representing the atomic sentences' truth values. The resulting four squares will represent the truth value for the formula. If the value assignment of the atomic sentences make the formula true, the square will be filled. If you are used to Punnet squares from genetics; truth is recessive for conjunctions and dominant for disjunctions, and filling the square signals the "phenotype" true being expressed.

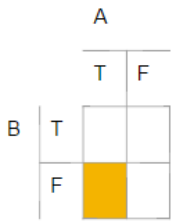


Figure 9: ***A*** ∧ ¬ ***B*** *The truth square clearly show that the clause A ∧ ¬ B is true if and only if A is true and B is false.*

For the same formula as before we will have the truth square as displayed in Figure 9. The atomic sentence A is represented by the top of the square, and B by the left side. As we discussed before A ∧ ¬ B is true if and only if A is true and B is false, so in the truth square the intersection of the T column and the F row will form a square that is filled.

## 5.2.Switching as Disjunctive Clauses

All the possible combinations of the switches we would want to do for the individuals A and B can be written as conjunctive clauses, of which A ∧ ¬ B used above is one. These clauses can be joined with disjunctions into the formula $\varphi$ = (A ∧ B) ∨ (A ∧ ¬ B) ∨ (¬ A ∧ B) ∨ (¬ A ∧ ¬ B) . All of these clauses were put into words in Table 2, and by joining them with disjunctions, 'or', we specify that one of these clauses must be true for the formula to be true.

Table 4: The truth table of the for the formula $\varphi$= ((A ∧ B) ∨ (A ∧ ¬ B)) ∨ ((¬ A ∧ B) ∨ (¬ A ∧ ¬ B))

| A | B | ((A∧ B) ∨ (A∧¬B )) ∨ ((¬A∧ B) ∨ (¬ A ∧ ¬B)) |
|---|---|---|
| T | T | T T T T T F F T T F T F T F F T F T |
| T | F | T F F T T T T F T F T F F F T F T F |
| F | T | F F T F F F F T T T F T T T F F F T |
| F | F | F F F F F F T F T T F F F T T T F |

Note that these disjunctions are inclusive; several of the clauses could be true at the same time according to this operator. However, by the nature of our problem assigning the atomic sentences values such that one clause is true will only be possible if all the others are false, and reciprocally if three clauses are false the fourth must be true.

For example if we do not (switch in both A and B), and we do not (switch in A but not B), and we do not (switch not in A but switch in B) then the only option left is to (switch not in A and not in B). There is an exclusive or operator, but as we have seen there is no need for one in this particular example and using an inclusive or will be helpful later on.

Furthermore, as all combinations of switching or not switching have been put into clauses, the formula will be true for any truth assignment of the literals A and B, as is visualized in Figure 10 and demonstrated in Table 4, where parentheses have been added to force the centermost or-operator to have the biggest scope.

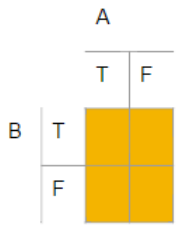


Figure 10: *The truth square for the formula $\varphi$= (A ∧ B) ∨ (A ∧ ¬ B) ∨ (¬ A ∧ B) ∨ ( A ∧B) shows that the formula evaluates to true for any value of the literals A and B*

As $\varphi$= (A ∧ B) ∨ (A ∧ ¬ B) ∨ (¬ A ∧ B) ∨ (¬ A ∧ ¬ B) represents all possible combinations of switches this is the formula we would like the program to formulate, assign weights and then use to find the optimal switch assignment using `toulbar2`. If the output from our max-SAT solver corresponds to A true and B false a switch would be made only in A.

However $\varphi$ is written in disjunctive normal form (DNF); the formula is a disjunction of clauses that only contain conjunctions of literals. max-SAT requires that the input formula is in conjunctive normal form (CNF) where the clauses only contain disjunct literals, and the clauses are joined by conjunction. (Barwise and Etchemendy 2002, Larrosa et al. 2008)

So the next step is how the DNF formula, from now on $\varphi_{DNF}$, can be written to a corresponding CNF formula $\varphi_{CNF}$ that will give the desired result from `toulbar2`. A good starting point is De Morgans laws which state that:

$$\neg(P \wedge Q) \leftrightarrow \neg P \vee \neg Q$$
$$\neg(P \vee Q) \leftrightarrow \neg P \wedge \neg Q$$

Combining DeMorgans laws with the fact that double negation is equivalent to no negation in this type of logic:

$$P \wedge Q \quad \text{(Premise)}$$
$$\neg(\neg(P \wedge Q)) \quad \text{(Double Negation)}$$
$$\neg(\neg P \vee \neg Q) \quad \text{(DeMorgan)}$$

So $P \wedge Q$ is equivalent to $\neg(\neg P \vee \neg Q)$. Removing the outer negation we get that the first and second statement are contradictory, when one is false the other is true, as demonstrated in Table 5. So there is a way to write a conjunction to its contradictory disjunction. If we do this for each of the conjunctions in $\varphi_{DNF}$ we can see an interesting pattern between one clause's conjunctive form and the other clauses' contradictory disjunctions, as in Figures 11, 12: if we join the contradictory disjunctions with a conjunction we'll have an equivalence to the conjunctive clause. Table 6 demonstrates that the formula A ∧ B is tautologically equivalent to ((A ∨ B) ∧ (¬ A ∨ B) ∧(A ∨ ¬ B)), and the same will hold for any other clause and set of clauses.

Table 5: The truth table of A ∧ B, ¬(¬ A ∨ ¬ B), and (¬ A ∨ ¬ B)

| A | B | ( A | ∧ | B ) | ¬ ( ¬ | A | ∨ | ¬ | B ) | ( ¬ | A | ∨ | ¬ | B ) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | T | T | T | T | T | F | T | F | T | F | T | F | F | T |
| T | F | T | F | F | F | F | T | T | F | F | T | T | T | F |
| F | T | F | F | T | F | T | F | T | T | T | F | T | F | T |
| F | F | F | F | F | F | T | F | T | F | T | F | T | T | F |

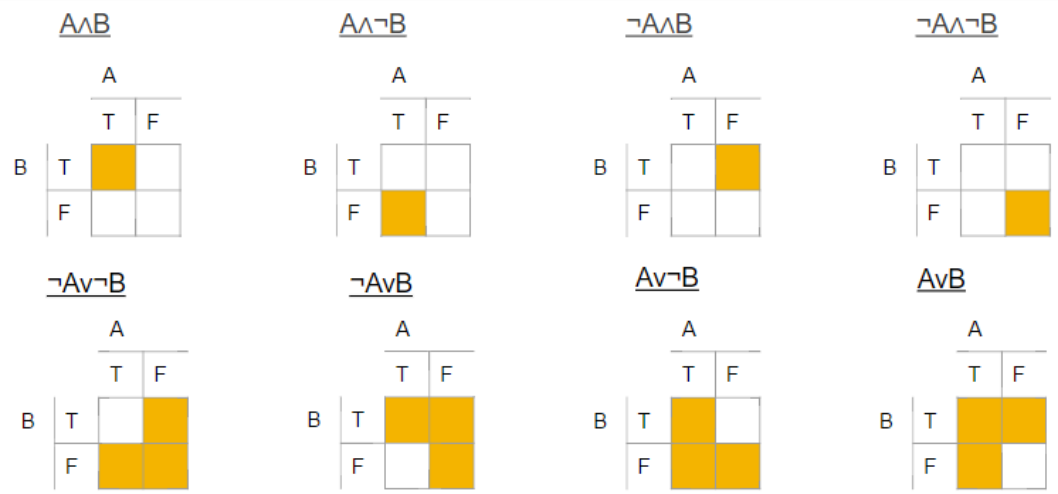


Figure 11: *The truth squares of all possible conjunction and disjunctions of A and B.*

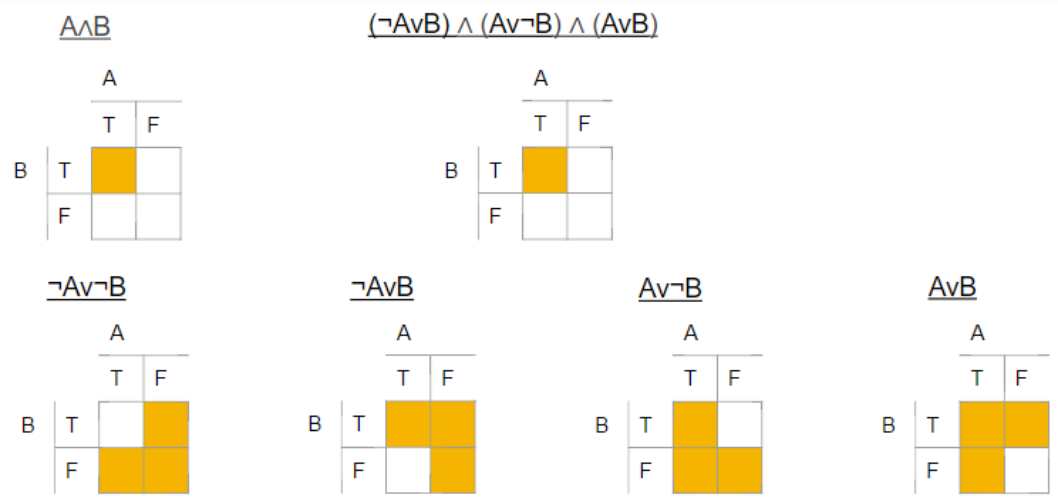


Figure 12: *The truth square for the conjunction the formula A ∧ B is equivalent to that of (A ∨ ¬ B) ∧ (¬ A ∨ B) ∧ (¬ A ∨ ¬ B), and the opposite of A ∨ B shows that the formula evaluates to true for any value of the literals A and B.*

So if we set both A and B to true, the conjunctive clause $A \wedge B$ is true, $\neg(\neg A \vee \neg B)$ is true, and $(\neg A \vee \neg B)$ is false. As demonstrated in Table 5 . The remaining disjunctions will be true, as demonstrated in Table 6.

Alternatively, think back to the fact that in $\varphi_{DNF}$ putting one clause to true was the equivalence of setting the remainder to false. If we join the corresponding contradictory disjunctions of $\varphi_{DNF}$ clauses into $\varphi_{CNF}$ with conjunctions we get $\varphi_{CNF}$ =(¬ A ∨ ¬ B) ∧ (¬ A ∨ B) ∧ (A ∨ ¬ B) ∧ ( A ∨ B) ∧ (A ∨ B). For this formula, if we set one clause to false the remaining clauses must be true. $\varphi_{CNF}$ is in CNF, and is therefore a formula we could feed into our solver, as long as we put weights on the clauses.

Table 6: The truth table of A ∧ B and ((A ∨ ¬ B) ∧ (¬ A ∨ B) ∧(¬ A ∨ ¬ B))

| A | B | A | ∧ | B | ( A | ∨ | ¬ | B ) | ∧ | ( ¬ | A | ∨ | B ) | ∧ | ( A | ∨ | B ) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | T | T | T | T | T | T | F | T | T | F | T | T | T | T | T | T | T |
| T | F | T | F | F | T | T | T | F | F | F | T | F | F | F | T | T | F |
| F | T | F | F | T | F | F | F | T | F | T | F | T | T | F | F | T | T |
| F | F | F | F | F | F | T | T | F | T | T | F | T | F | F | F | F | F |

## 5.2.1.Weighted Clauses in Formulas

If A∧B is true and carries weight it follows that the corresponding true disjunctions in conjunction carries the same weight.

In weighted max-SAT the weight corresponds to the cost of violating a clause (Larrosa et al. 2008); the smaller the weight is for a clause, the more likely it is that the clause will be unsatisfied. The starting point of `cnF2freq` already lends itself to calculating weights to all clauses in $\varphi_{DNF}$ with a small tweak, and there the bigger weight means that the clause is more likely to be true.

A clause that is likely to be true in $\varphi_{DNF}$ should be correspondingly likely to be false in $\varphi_{CNF}$ so all that is needed is to make a corresponding transformation of the weights. Intuitively if $w_i$ is the weight of a conjunctive clause we can put that the weight of the contradictory disjunctive clause is $-w_i$. However, since the relative difference between the clauses remain the same if we add a constant, we can for the ease of implementation put that the weight of a disjunctive clause is $K - w_i$ where $K \geq \max(w_i)$.

So the set of clauses $\{\{A \wedge B, w_1\}, \{A \wedge \neg B, w_2\}, \{\neg A \wedge B, w_3\}, \{\neg A \wedge \neg B, w_4\}\}$ corresponds to the set $\{\{\neg A \vee \neg B, C - w_1\}, \{\neg A \vee B, C - w_2\}, \{A \vee B, C - w_3\}, \{A \vee B, C - w_4\}\}$. The first set can be used within `cnF2freq` and the second when interacting with the weighted max-SAT solver which for now is `toulbar2`.

As we will use every possible assignment to our literals in our set of clauses, allowing all possible switches, we know that if one conjunctive clause is true, the rest are false. If we negate its corresponding disjunct derived by De Morgans law that disjunct is false. Correspondingly, the negation of all the corresponding disjuncts to the remaining conjunctions will be true.

# 6.Results

This project was divided into two main parts. Formulating a solution, and then implementing it. The formulation and the theoretical reasoning that shows that the solution is viable have been covered in previous sections. To summarize; we can translate switching possibilities, including multiple dependent switches, into a set of logical clauses. These clauses can then be reformulated to serve as input to preexisting solvers. The weight assigned to each clause can be derived from the weights already calculated by `cnF2freq`.

After finding the theoretical solution, the next intuitive step is solving the problem in practice.

## 6.1.Implementation

In order to implement the theoretical solution, the program needed to be able to do a correct logical translation of existing evaluations, group the resulting clauses, produce input files for `toulbar2`, call `toulbar2`, and interpret the solver's output into a format that could be read by the existing code in `cnF2freq`. This was done in a fork of `cnF2freq`'s Plantimpute_modern branch. In addition to this multiple dependent switches had to be allowed. This last part was realized by removing the right checking point from the code.

The added code adheres to the C++11 standard. The logical clauses were hard coded for a maximum of three generations, and the clauses were stored in a new structure to be iterated over when writing the input file.

As described in Methods a wcnf input format is needed, and from Theory we know that if we generate weights, $w_i$ for the switches expressed as conjunction, we can find their corresponding negated disjunctions and set their weights to $C - w_i$ where $C$ is a constant. Based on this an input file to `toulbar2` is created, the solver is run, its output read, and translated so that its results can be used within `cnF2freq`.

The new `cnF2freq` branch can perform both single switches, no switches and multiple dependent switches. The goal of this project has thus been met.

The output format from `toulbar2` deviates in its current implementation from Evaluation of MaxSAT Solvers standard. Thus if another solver is to be fitted into `cnF2freq` the interpretation of the solvers output will likely need to be rewritten.

### 6.1.1.Assumptions

The added code assumes the following:

- A maximum of 3 generations will be considered part of the focus tree.
- No incest within 3 generations.
- An individual only has a grandparent if the individual has a parent that has a parent.
- An individual can have a maximum of 2 biological parents.
- Each parent contributes to the offspring with one haplotype.

# 7.Discussion

The goal of this project was to implement a solution that allows `cnF2freq` to perform multiple dependent switches as well as single or no switches.

Roughly half of this project was spent on formulating the needs of the solution and finding a solution that could be implemented. At the start I tried to formulate it as a graph problem, but I found it easier to think of it in a first-order logic setting. It is quite fitting that the max-SAT problem can be thought of as a graph-edge problem, but is equally a logical formula solution problem Larrosa et al. (2008).

After formulating the problem as a logical formula the next problem was to translate it to the right type of formula for a solver, going from DNF to CNF. The logical soundness of the translation of the problem has been demonstrated for a small example in the Theory section.

After I handed over my code to my supervisor, having worked on it for as long as the project would allow, some minor adjustments to the code needed to be made for it to run on a separate Unix version. There were also some minor corrections needed.The updated fork can now be run on the same data as `cnF2freq` could be at the start of this project, but with the possibility of doing multiple dependent switches. To find if this function improves the found haplotypes, and how often these problems of the program generating biologically unsound haplotypes on real data actually occur further studies should be made.

If this type of solution is deemed useful, it would be of interest to do an evaluation of which max-SAT solver would be best suited for `cnF2freq`. To further improve the already implemented changes the new solution should be changed to directly call `toulbar2`'s functions instead of running the program.

The result of this project is a theoretical solution for allowing multiple dependent switches in cnF2freq, and an almost production ready practical solution.

# 8.Acknowledgments

I would like to thank my supervisor Carl Nettelblad for his guidance, helpful explanations, quick feedback and energetic optimism during this project. I would also like to thank Lisa Klasson, my subject reader for her advice on so much more than the subject matter of this thesis; without Lisa this journey would not have been as smooth.

I am grateful for my friends and family, for their support throughout my education and this project. Not in the least Therese and Marie for attempting to understand a bioinformatic degree project and providing feedback on my popular science summary; Julia for all her calming, encouraging words; my brother for his late night attempt at spell-checking as well as his constant faith that finishing would not be an issue; and not in the least my parents for their endless support.

In addition, my thanks to everyone that I came in contact with through this project for their patience with me as I have endeavoured to find, write, and finish this degree project.

# References


Abecasis, G. R., Cherny, S. S., Cookson, W. O., and Cardon, L. R. (2002). Merlin–rapid analysis of dense genetic maps using sparse gene flow trees. *Nature Genetics*, 30(1):97–101.

Barwise, J. and Etchemendy, J. (2002). *Language proof and logic: [text/software package]*. CSLI Publications.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Browning, B. L. and Browning, S. R. (2009). A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. *The American Journal of Human Genetics*, 84(2):210–223.

Browning, S. R. and Browning, B. L. (2011). Haplotype phasing: Existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714.

Campbell, N. A. and Reece, J. B. (2010). Fertilization and meiosis alternate in sexual life cycles. In *Biology*, pages 250–253. Pearson/Cummings, 8. ed., pearson international ed., [nachdr.] edition.

Chen, R., Mias, G., Li-Pook-Than, J., Jiang, L., Lam, H., Chen, R., Miriami, E., Karczewski, K., Hariharan, M., Dewey, F., Cheng, Y., Clark, M., Im, H., Habegger, L., Balasubramanian, S., O'Huallachain, M., Dudley, J., Hillenmeyer, S., Haraksingh, R., Sharon, D., Euskirchen, G., Lacroute, P., Bettinger, K., Boyle, A., Kasowski, M., Grubert, F., Seki, S., Garcia, M., Whirl-Carrillo, M., Gallardo, M., Blasco, M., Greenberg, P., Snyder, P., Klein, T., Altman, R., Butte, A. J., Ashley, E., Gerstein, M., Nadeau, K., Tang, H., and Snyder, M. (2012). Personal omics profiling reveals dynamic molecular and medical phenotypes. *Cell*, 148(6):1293–1307.

Gibson, G. and Muse, S. V. (2009). *A primer of genome science*. Sinauer Associates, 3. ed edition.

Hurley, B., O'Sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., and Givry, S. d. (2016). Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21(3):413–434.

Jones, N. C. and Pevzner, P. (2004). In *An introduction to bioinformatics algorithms*, Computational molecular biology. MIT Press.

Larrosa, J., Heras, F., and de Givry, S. (2008). A logical approach to efficient max-SAT solving. *Artificial Intelligence*, 172(2):204–233.

Narodytska, N. and Bacchus, F. (2014). Maximum satisfiability using core-guided MaxSAT resolution. In *AAAI*, pages 2717–2723.

Nettelblad, C. (2011). Haplotype inference based on hidden markov models in the QTL-MAS 2010 multi-generational dataset. *BMC Proceedings*, 5:S10.

Nettelblad, C. (2012a). Inferring haplotypes and parental genotypes in larger full sib-ships and other pedigrees with missing or erroneous genotype data. *BMC Genetics*, 13:85.

Nettelblad, C. (2012b). *Two Optimization Problems in Genetics Multi-dimensional QTL Analysis and Haplotype Inference*. Acta Universitatis Upsaliensis. OCLC: 939834304.

Nettelblad, C., Holmgren, S., Crooks, L., and Carlborg, O. (2009). cnF2freq : Efficient determination of genotype and haplotype probabilities in outbred populations using markov models. *Bioinformatics and Computational Biology*, pages 307–319.

O'Connell, J., Gurdasani, D., Delaneau, O., Pirastu, N., Ulivi, S., Cocca, M., Traglia, M., Huang, J., Huffman, J. E., Rudan, I., McQuillan, R., Fraser, R. M., Campbell, H., Polasek, O., Asiki, G., Ekoru, K., Hayward, C., Wright, A. F., Vitart, V., Navarro, P., Zagury, J.-F., Wilson, J. F., Toniolo, D., Gasparini, P., Soranzo, N., Sandhu, M. S., and Marchini, J. (2014). A general approach for haplotype phasing across the full spectrum of relatedness. *PLoS Genetics*, 10(4):e1004234.

Snyder, M. W., Adey, A., Kitzman, J. O., and Shendure, J. (2015). Haplotype-resolved genome sequencing: experimental methods and applications. *Nature Reviews Genetics*, 16(6):344–358.

Talenti, A., Bertolini, F., Pagnacco, G., Pilla, F., Ajmone-Marsan, P., Rothschild, M. F., and Crepaldi, P. (2017). The valdostana goat: a genome-wide investigation of the distinctiveness of its selective sweep regions. *Mammalian Genome*, 28(3):114–128.

Wang, W. Y. S., Barratt, B. J., Clayton, D. G., and Todd, J. A. (2005). Genome-wide association studies: theoretical and practical concerns. *Nature Reviews Genetics*, 6(2):109–118.

Xing, Z. and Zhang, W. (2005). MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability. *Artificial Intelligence*, 164(1):47–80.

Zvelebil, M. J. and Baum, J. O. (2008). 6.2 profile hidden markov models. In *Understanding bioinformatics*, pages 179–196. Garland Science.