ECM3401: Individual Project and Review

# Optimising the Results of a Genome Wide Association Study Using Machine Learning

**Raoul Dias - 660064979**

**Abstract**

**Keywords:** Machine Learning, Genome-Wide Association Study, Stochastic Search, Ant-Colony Optimisation, Tabu List, Implicit Causal Models, Neural Networks, Deep Learning, Optimisation

This project looks into the machine learning and optimisation methods used to find gene-gene interaction in a Genome-Wide Association Study(GWAS). These association are found by measuring differences in genetic variance(Single Nucleotide Polymorphisms) between cases and controls, in a GWAS. The project consists of three development chunks, the data representation, the objective functions and the algorithms. This project looks at two stochastic search algorithms, Ant-Colony Optimisation and Ant-Colony Optimisation with Tabu List, and one deep learning generative causal model, The Implicit Causal Model. The paper mainly focuses on finding two locus analyses in the Type-II diabetes dataset.

*I certify that all the work in this dissertation that is not my own has been identified*

## Contents

# 1 Introduction

A decade and a half ago, an international research team completed an ambitious effort to read the 3 billion letters of genetic information found in every human cell. The program, known as the Human Genome Project, provided the blueprint for human life, an achievement that has been compared to landing a man on the moon. This opens the doors to some incredible opportunities for us to identify, diagnose and treat some of the diseases that we have long considered incurable.[1]

Recent advances in sequencing technology has allowed researchers to sequence the genomes of thousands of individuals and to compare genomes across a large cohort of subjects. Such studies, known as genome-wide association studies (GWAS), capture the small variations in genomes among individuals and attempt to understand the association between these and the variation in phenotypic traits.

A Genome Wide Association Study (GWAS) is an observational study of a genome-wide set of genetic variants in different individuals, in order to see if any variant is associated with a trait. GWA studies capture the small variations in genomes (known as single nucleotide polymorphisms (SNPs) [2] ) among individuals and attempt to understand the association between these and the variation in phenotypic traits such as height, body mass index and the propensity to develop certain diseases. [3][4]

GWAS studies came into the fray of genetic epidemiology around 2007, when a GWAS was conducted to find the genetic causes of type II diabetes. Since then many other disease datasets have been created from large projects such as the Wellcome Trust Case Control Consortium (WTCCC) [5] and UK Biobank [6]. GWAS gives us the opportunity to hack our genes and find the genetic basis for diseases. This give us the potential to provide early treatment and planning for patients with these diseases. [7]

Associations between SNPs and a disease can be found by iteratively exploring the association of each SNP in turn, a computationally complex but feasible problem. The exploration of associations between more than one SNP and a disease is an even more computationally complex problem. In Figure 1 we can see how the computational time and number of checks required when a higher number of genes are being explored. Gene-gene interactions can be investigated as an additive model where the effects of possessing a combination of SNPs are added together, or through other mechanisms like Epistasis where the individual effect of each SNP might be small, but in combination, the effect is large. [3] The aim of this project will be to try to discover epistatic interactions within the dataset.

## 1.1 Computational Motivation

From a computational perspective, GWAS present a significant challenge as there are hundreds of thousands of SNPs (variables) per individual. Any computational approaches used for analysis therefore must be able to scale up. Many examples [3] of GWAS data analysis exist in the literature that successfully demonstrate the association between a single SNP and phenotypic traits such as diseases. Single SNPs that are associated with a disease are called simple Mendilian traits [8], and can be found by modern computers in reasonable computational time. However this computational time increases significantly when the task is switched to complex traits that rely on a combination of many SNPs acting in tandem. [7]

Our specific GWAS consists of over 500,000 SNPs per individual and more than 5,000 individuals, meaning that we collectively have around 2.5 billion unique data points. Single-locus analysis has provided us with know associations for diseases such as type-2 diabetes and traits such as height. However, these SNPs alone are not enough to find a genetic basis for the phenotypic traits. There is a considerable amount of missing heritability. [7] For example for the phenotypic trait height, only approximately 10% of variation in height can be explained by traditional single locus analysis. [7] This missing heritability can be explained by combinations of SNPs(gene-gene interactions) . Standard single-locus analysis can be carried out in a computationally efficient time, using software packages like PLINK. [7]

When we extended these analyses to analyse combinations of SNPs, the computational burden becomes incredibly large, leading to the development of a variety of optimisation techniques to overcome this burden. [9]. The approaches can be divided into two broad categories, approaches that pre-scan the GWAS for single-point associations, and then exhaustively search this reduced dataset(known as filter approaches), and those that use heuristic search techniques to search the whole set of data for multiple associations( known as a wrapper approach).[7]

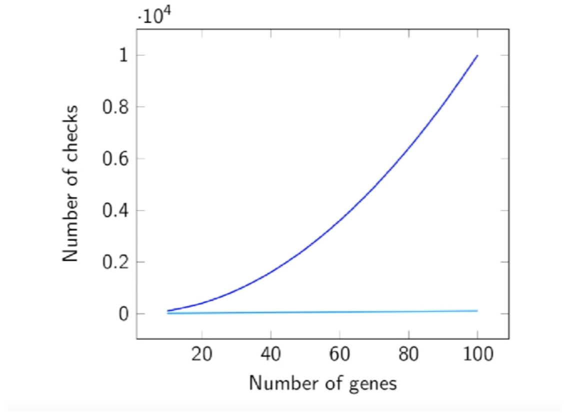


Figure 1: Number of checks required as the number of SNPs increase

## 1.2 Biological Motivation

Type II diabetes(T2D) is a metabolic disorder that results in hyperglycemia. Hyperglycemia, or raised blood sugar, is a common effect of uncontrolled diabetes, and over time leads to serious damage to many of the body's systems, especially the nerves and blood vessels. The disease effects more than 422 million people worldwide [8]. In the UK alone diabetes affects about 6% of the adult population, and this figure is expected to rise in the coming years.[10] The Genetic combinations that could potentially be uncovered by the analysis of a GWAS, could lead to the early discovery of people predisposed to T2D. This could offer early treatment options and lifestyle changes that could potentially save lives. Furthermore, the pharmaceutical industry can also use this information to design drugs that target specific genetic targets.[11]

## 1.3 Overall Goals

The overall goals for the project will be to:

1. Investigate the methodologies and models that have been presented.
2. Implement these models and test them on the Type-II Diabetes dataset.
3. Test their feasibility and efficiency in finding gene-gene interactions.

# 2 Summary of Literature and Project Specification

## 2.1 Literature Review

This subsection provides a brief summary of the concepts discussed in the literature review.

### 2.1.1 Single Nucleotide Polymorphisms

Single nucleotide polymorphisms, frequently called SNPs (pronounced “snips”), are the most common type of genetic variation among people. Each SNP represents a difference in a single DNA building block, called a nucleotide. For example, a SNP may replace the nucleotide cytosine (C) with the nucleotide thymine (T) in a certain stretch of DNA. [12]

### 2.1.2 Epistasis

Epistasis can be broadly defined as the cumulative effect of multiple genotypic variations on a phenotypic trait. [13] In the literature ,[14] there are several contrasting definitions of epistasis. Functional epistasis is a functional descriptor that addresses molecular interactions between SNPs. [15] Another definition of epistasis, is Compositional Epistasis, which is defined as the blocking of one allelic effect by another allele at a different locus [15]. Statistical epistasis, is defined as the statistical deviation from the additive effects of two loci on a phenotype [15]

### 2.1.3 Statistical Hypothesis Tests

Statistical Hypothesis Tests are used to test whether the associations, seemingly found when a locus of SNPs is observed for the group of Cases(the hypothesis), are valid, and not just a matter of chance(e.g., Pearson’s Chi-Squared Test). These tests produce a value of statistical power, which is a measure of how strong the proposed hypothesis is. These tests can be used as an objective function in our program and as a measure of fitness for our models. [16]

### 2.1.4 Defining Association as a Classification Problem

Statistical interaction between two domains can be described by a linear model describing the relationship between some outcome variable and some predictor variables. We propose a particular model for how we believe the predictors might relate to the outcome, and we use data (i.e. measurements of the relevant variables on a number of individuals) to determine how well the model fits our observed data, and to compare the fit of different models. [17]

### 2.1.5 Filter

Filter approaches work by pre-selecting a group of candidate SNPs, based on their single-association p-values, and then performing an exhaustive search of the reduced dataset. In the filtering stage, the SNPs with the highest marginal values are chosen, and then combined and further compared in combination to test for gene-gene interaction. This can be problematic when testing for epistasis, as genes that might not have large marginal effects, might still contribute to epistasis, as most complex traits consist of a range of SNPs with different contributing factors, but in combination can explain the heritability of the trait. Therefore, filtering might omit some SNPs that are essential to the association with the trait. The advantage of the filter approaches are that they are generally quicker than wrapper approaches, due to the reduced final set of SNPs. [18]

### 2.1.6 Wrapper

Wrapper approaches use a global search technique, that is able to search the space of all combinations of SNPs. Wrapper approaches can therefore theoretically find higher-order interactions between SNPs, that filter approaches would not be able to find. Although, the stochastic nature of wrapper algorithms mean that they are not guaranteed to find optimal solutions.

### 2.1.7 Decision Trees and Random Forest

Decision Trees can be used to explicitly represent decisions and the decision making process. They create a model to preform classification by splitting variables based on their attributes. The RF method was used by Lunetta et al in [19] as a screening procedure to identify a set of risk-associated single SNPs from the large number of unassociated SNPs of complex disease model.

### 2.1.8 Neural Networks

A Neural Network is a machine learning algorithm that is inspired by the neural structure of the biological human brain. [20] They are made up of layers that transfer values to each other through nodes. Neural networks are a flexible statistical tool to model any functional relationship between covariates and response variables. Therefore, they represent a promising approach to deal with the difficulties associated with modeling biological gene-gene interactions.[21]

### 2.1.9 Ant-Colony Optimisation

Ant-Colony Optimisation is a stochastic optimisation algorithm that is based on the the behaviour of ants in nature. When an ant finds a source of food, it walks back to the colony leaving ”markers” (pheromones) that show the path has food. When other ants come across the markers, they are likely to follow the path with a certain probability. If they do, they then populate the path with their own markers as they bring the food back. As more ants find the path, it gets stronger until there are a couple streams of ants traveling to various food sources near the colony. The pheromone markers fade as time passes.[22] An Ant-Colony optimisation method was proposed by Sapin et al, that used a tournament selection method to select a random set of SNPs and test their pairwise interactions using a chi-squared metric as a fitness function, giving pheromone to the best associated pairs. This process was repeated over several epochs. After the model was run over multiple epochs, the SNPs with the highest amount of pheromone, were theorised to be associated to the diseases, in this case type-2 diabetes. [7]

### 2.1.10 BOOST

BOolean Operation-based Screening and Testing(BOOST), is a current fast filter-based approach for the analysis of complex traits in GWAS studies. [7] The BOOST system designed a Boolean representation of the genotypic data, which promotes space and CPU efficiency as it involves only Boolean values and allows for the use of fast bitwise operations to obtain contingency tables. The search process consists of two steps: a screening and a testing stage. In the screening stage, a non-iterative method is used to approximate the likelihood ratio (a form of statistical hypothesis test) statistic for all pairs of SNPs, and then selecting the SNPs that pass a particular threshold value. Most insignificant SNPs are filtered out, and the surviving SNPs then applied to the testing phase. The testing stage uses a classical likelihood ratio test to measure the interaction effects of the selected pairs of SNPs. [23]

## 2.2 Project Specification

The development of the project can be broadly divided into three chunks:

1. The Representation of the SNP arrays into usable form
2. The Objective Function for the calculation of significance of associations
3. The algorithms to find the epistatic/gene-gene associations in the dataset

### 2.2.1 Data Representation

The PLINK format stores the gnomic data in three binary encoded files (.bed,.bim and .fam). These files will have to be converted to a format that can be used by our algorithms for analysis. A possible representation would be in the form of NumPy arrays. A program would be needed to convert the binary encoding to NumPy arrays. NumPy gives Python access to quick multidimensional arrays that we use to efficiently store the SNP data, giving Python Matlab-like functionality. Calculations on NumPy arrays also run at C++ speeds, meaning the calculations are much faster than the ordinary Python equivalent. The NumPy arrays can be modeled either based on SNPs or Individuals based on the model. The development required will be to turn the PLINK format into an appropriate representation for the algorithms to use.

### 2.2.2 Objective Function

We will be experimenting with a few statistical hypothesis tests for association as our objective functions. One example would be the Chi-Squared Test, which would require some development to coalesce the data and then could use the SciPy package in python for the implementation of the test statistic. The implementation of the objective functions will be done in the python programming language.

### 2.2.3 Algorithms

We are going to be using an Ant-Colony Optimisation method, A Deep Feed Forward Neural Network and experimenting with a Convolutional Neural Network to find our associations. These algorithms will be implemented in python, with the help of the NumPy, SciPy, SciKit-Learn and Tensorflow libraries, along with some parallel processing libraries, potentially making use of multiprocessing libraries for either CPU or GPUs for calculations.

## 3 Design

This section will provide an overview of the design of the final project. A more in-depth analysis of the individual components, as well as the challenges faced during implementation, will be outlined in later sections.

### 3.1 Flow Chart

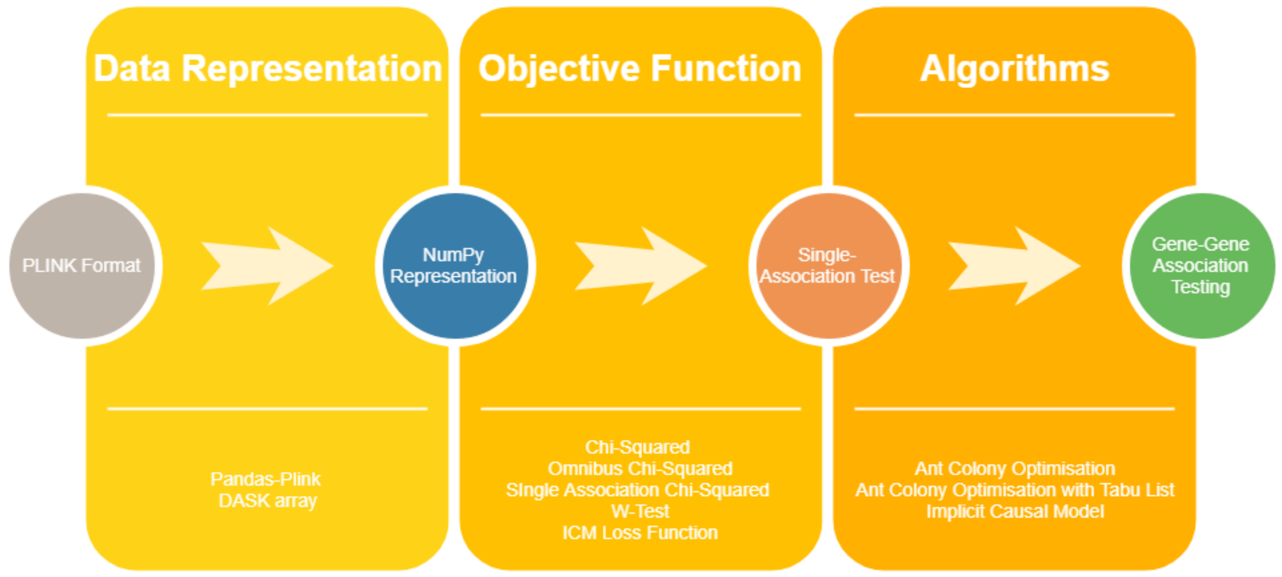


Figure 2: Overall Project Flow Chart

### 3.2 Data representation

The Data Representation in the project has been implemented as a python package, which is used to export the gnomic data into a Python environment. The package uses a C back-end to effectively parse through the .bed binary file.

The package then uses the C Foreign Function Interface package in Python to move the data into a python environment. After which, the resulting data is stored in DASK arrays, which are a collection of interlinked NumPy style arrays, that use lazy loading (loading exactly which parts of the dataset are required just in time) to overcome the large size of the dataset.

The bim and fam files, which contain the data that corresponds to the SNPs and individuals in the dataset are converted to pandas data frames, which allow quick and reliable access to the data, that will be used in conjunction with the bed DASK array in the objective function and algorithms. The phenotype file, a text file, which has the Type II diabetes status, as well as other phenotypic information, such as BMI, is converted into a pandas dataframe.

### 3.3 Objective Functions

There have been a few different Objective functions that have been developed in the code, namely, a 2x3 $\chi^2$ test for single association analysis, a 2x8 $\chi^2$ omnibus test , as well as a partitioned $\chi^2$ test for the ant colony optimisation models. All the objective functions have been developed in python, using the NumPy and SciPy packages, namely the chi2_contingency function from SciPy, which take a contingency table as input and outputs the $\chi^2$ statistic, p-value, degrees of freedom and expected frequencies of the data. The contingency tables are therefore calculated by parsing the database, and then put into the chi2_contingency function.

## 3.4 Algorithms

The algorithms that have been implemented are as follows: Single Association Analysis, Ant-Colony Optimisation (ACO), Ant-Colony Optimisation with Tabu List and an Implicit Causal Model (ICM). All four models have been implemented on IPython (Jupyter) notebooks, due to their portability, ability to add markup, and their ability to run separate chunks of code, which is needed in this project for testing and evaluation of the models.

The single Association analysis is implemented as a python class, and the $\chi^2$ are parallelised using the multiprocessing package on python to minimise the run time. A function has also been implemented to generate a Manhattan plot, using the single association p-values.

The Ant-Colony Optimisation and Tabu list models have been implemented as classes on python. The ants have an included option of parallelisation, which is implemented using the Multiprocessing library on python, which significantly decreases the time taken to run the models.

The Implicit Causal Model, which consists of two neural networks, is written using Edward, which is a Python library for probabilistic modeling, inference, and criticism, as well as tensorflow, which is an end-to-end open source platform for machine learning. The neural networks are built using tensorflow, and the general Bayesian inference is done using the Edward library.

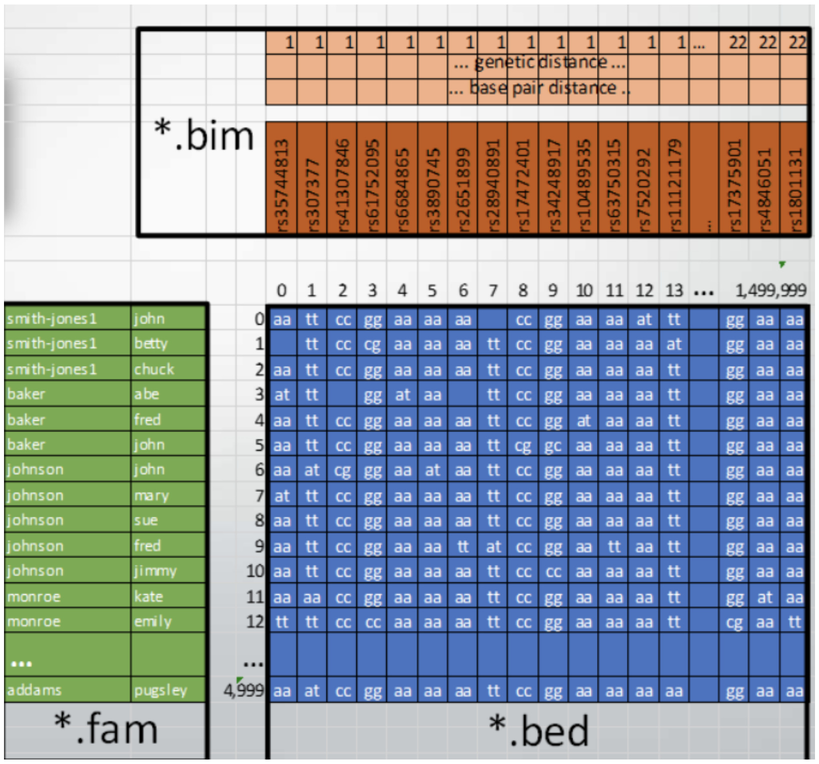
## 4 Data

This section will be an overview of the data that we will be using in the project. The Dataset that we are using is from the Exeter 10,000 (EXTEND) cohort, which was a set of Genome-Wide Association Studies done on people in and around the Exeter region, to test for diseases such as diabetes, heart disease and dementia [24]. Our particular GWAS, was based on relation to the Type-II diabetes (T2D) phenotypic trait, and consists of 10,000 individuals and nearly 500,000 SNPs.

This section will attempt to achieve the following objectives:

- Examine the structure of the data
- Explain the pre-processing done on the data
- Contrast the various possible data representations
- Justify the chosen data representation
- Elucidate the program structure of the data representation component

### 4.1 Structure of the data



| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | ... | 22 | 22 | 22 |
| | | | rs35744813 | rs307377 | rs41307846 | rs61752095 | rs6684865 | rs3890745 | rs2651899 | rs28940891 | rs17472401 | rs34248917 | rs10489535 | rs63750315 | rs7520292 | rs11121179 | ... | rs17375901 | rs4846051 | rs1801131 |
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | ... | | | 1,499,999 |
| smith-jones1 | john | 0 | aa | tt | cc | gg | aa | aa | aa | | cc | gg | aa | aa | at | tt | | gg | aa | aa |
| smith-jones1 | betty | 1 | | tt | cc | cg | aa | aa | aa | tt | cc | gg | aa | aa | aa | at | | gg | aa | aa |
| smith-jones1 | chuck | 2 | aa | tt | cc | gg | aa | aa | aa | tt | cc | gg | aa | aa | aa | tt | | gg | aa | aa |
| baker | abe | 3 | at | tt | | gg | at | aa | | tt | cc | gg | aa | aa | aa | tt | | gg | aa | aa |
| baker | fred | 4 | aa | tt | cc | gg | aa | aa | aa | tt | cc | gg | at | aa | aa | tt | | gg | aa | aa |
| baker | john | 5 | aa | tt | cc | gg | aa | aa | aa | tt | cg | gc | aa | aa | aa | tt | | gg | aa | aa |
| johnson | john | 6 | aa | at | cg | gg | aa | at | aa | tt | cc | gg | aa | aa | aa | tt | | gg | aa | aa |
| johnson | mary | 7 | at | tt | cc | gg | aa | aa | aa | tt | cc | gg | aa | aa | aa | tt | | gg | aa | aa |
| johnson | sue | 8 | aa | tt | cc | gg | aa | aa | aa | tt | cc | gg | aa | aa | aa | tt | | gg | aa | aa |
| johnson | fred | 9 | aa | tt | cc | gg | aa | aa | tt | at | cc | gg | aa | tt | aa | tt | | gg | aa | aa |
| johnson | jimmy | 10 | aa | tt | cc | gg | aa | aa | aa | tt | cc | cc | aa | aa | aa | tt | | gg | aa | aa |
| monroe | kate | 11 | aa | aa | cc | gg | aa | aa | aa | tt | cc | gg | aa | aa | aa | tt | | gg | at | aa |
| monroe | emily | 12 | tt | tt | cc | cc | aa | aa | aa | tt | cc | gg | aa | aa | aa | tt | | cg | aa | tt |
| ... | | ... | | | | | | | | | | | | | | | | | | |
| addams | pugsley | 4,999 | aa | at | cc | gg | aa | aa | aa | tt | cc | gg | aa | aa | aa | aa | | gg | aa | aa |

Figure 3: PLINK structure [25]

The data is stored in the PLINK format. The PLINK binary encoding format consists of three types of files, .bed, .bim and .fam. The .bed file contains the genotypic data. For the genotype data, each byte encodes up to four genotypes (2 bits per genotype). The coding is:
00 Homozygote

01 Heterozygote
11 Homozygote
10 Missing genotype
Each bit in the .bed file is read from right to left. The .bim file is the extended map file, which also includes the names of the alleles: (chromosome, SNP, cM, base-position, allele 1, allele 2). The .fam file has the unique identifiers for the individuals, as well as the case/control status. The phenotype file contains a list of phenotypic traits i.e., age, BMI and Type II diabetes status, and is represented in the form of a text file.

The actual binary data are blocks of 8 bits (a byte) in the center: the first 3 bytes have a special meaning. The first two are fixed, a 'magic number' that enables PLINK to confirm that a BED file is really a BED file. That is, BED files should always start 01101100 00011011. The third byte indicates whether the BED file is in SNP-major or individual-major mode: a value of 00000001 indicates SNP-major (i.e. list all individuals for first SNP, all individuals for second SNP, etc) whereas a value of 00000000 indicates individual-major (i.e. list all SNPs for the first individual, list all SNPs for the second individual, etc.)

### 4.2 Pre-Processing of Data

There are a variety of exclusion criteria that have to be applied to GWAS data before they can be processed. These include:

- The Hardy-Weinberg Equilibrium(HWE) Exact Test, which is a measure of population stratification.
- Minor Allele Frequency(MAF), which is the frequency at which the second most common allele occurs in a given population.
- Missing Data Proportion(MDP), which is the percentage of missing data in the dataset.

There is further information about these criteria, readers are referred to the GWAS literature for further information. [26]

The SNPs that are kept are those meeting the following criteria:

- Hardy-Weinberg Equilibrium(HWE) Exact Test 25.7e-7
- Minor Allele Frequency 21%
- Missing Data Proportion 15%

The remaining data after the application of these criteria contain 479,000 SNPs for the T2D study.

### 4.3 Data Representations

The GWAS dataset is extremely large, 500,000 SNPs and 10,000 individuals, totalling 5 billion total discrete variables. If the data was stored as standard floating points in arrays (each floating point = 4 bytes), the amount of memory needed to store the whole dataset would be 20 GB. It is therefore paramount to chose an appropriate representation that is able to accommodate the large size of the data, quick enough to enable the algorithms to run in reasonable computational time, and robust enough for the algorithms to use without a massive confessional overhead. In this section, we will examine a few possible representations that could potentially solve these issues.

### 4.3.1 Binary Encoding

The first avenue that we could consider would be to try to use some form of compression to store the whole dataset. One possible method is to binary encode the dataset, a possible encoding of SNPs would be:

- Unknown: 0
- Common Homozygous(CH):1
- Heterozygous(h):2
- Rare Homozygous(RH):3

If we raise the SNP number to the power of the encoding above, four SNPs can be stored in one byte.

If we used this form of lossless compression, the size of the database would be reduced four-fold, which would enable it to be stored on RAM. This representation would provide us the opportunity to access the data directly from RAM, although it would require extra processing to compress and decompress the data. This representation would also be challenging to use with some of our models due to its representational capacity.

### 4.3.2 DASK Array

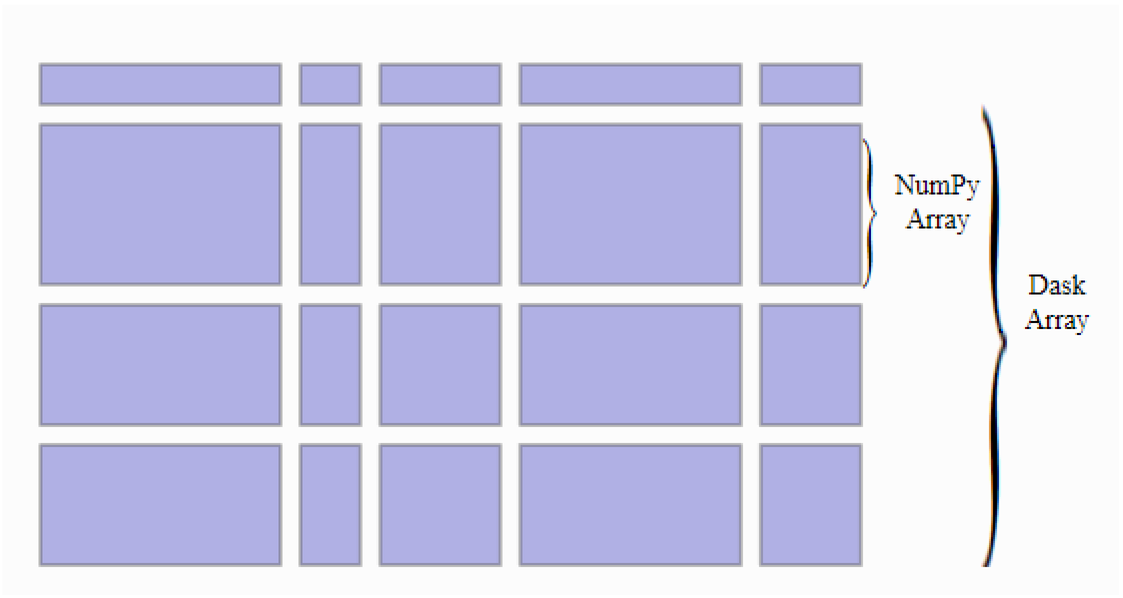


Figure 4: DASK array structure

The other potential option for representation is to split up the array into multiple smaller arrays and store some on disk. There is an extra overhead caused by paging, that would outweigh the extra processing required to implement the binary representation, therefore any proposed solution down this avenue would need an intelligent work around to the extra time taken due to the paging. One possible solution is to use DASK arrays to overcome the spatial and computational burden caused.

DASK Array implements a subset of the NumPy ndarray interface using blocked algorithms, cutting up the large array into many small arrays. This allows us compute on arrays larger than memory using all of our cores. These blocked algorithms are coordinated using DASK graphs.[27]

In Figure 4 we can see a DASK array with a number of NumPy arrays that are divided into chunks of smaller NumPy arrays, this is useful for the project as it allows us to use the representational and computational power of the NumPy ndarray on our large dataset. Choosing the correct chunk size can effect the computational time of our program, for example, using chunks of whole SNP arrays would allow for quick computation, as our objective function only works on a few SNPs at a time, and therefore would only need an initial load, the bulk of the computation done by the objective function would be parsing though this array, and forming and calculating contingency tables, which would be made much quicker and computationally efficient by using chunks that are a SNP large. The DASK array stores as many chunks on memory as possible and then uses the DASK graph, which is an intricate multiprocessing scheduler to coordinate the quick loading of the required next chunks. The use of lazy loading means that the individual chunks are loaded only exactly when they are needed by the algorithms, significantly cutting down the memory constraints.

The ability to use NumPy arrays will allow the models to use quick matrix multiplication, that will increase the computational power of the data. The DASK arrays also allow room for large scale-ability as the chunks don't necessarily have to be stored on the same machine, as they are built to be run on clusters and over large file systems, this would help immensely if we were to scale the models to run on clusters. Essentially, it enables us to write code once and then choose to either run it locally or deploy to a multi-node cluster using a just normal Pythonic syntax.[28]

#### 4.3.3 Conclusion

Therefore, for our models that don't require the representational capacity of the NumPy arrays, i.e., our Ant Colony Optimisation models, we can use the binary encoding representation, and for the models that could benefit from the NumPy functionality and would also benefit from the options of scalability, so that we can run them on clusters, i.e., the Implicit Causal Model, we can use the DASK array implementation.

### 4.4 Program Structure

The python package that was designed consists of five main components:

1. The C Program to read the bed file.
2. Transferring the data to a Python environment.
3. The Python Program to convert the representation into a DASK array.
4. The Python Program to convert the representation into the binary encoding.
5. Converting the .bim,.fam and phenotype file into pandas dataframes.

The package uses a C back-end to effectively parse through the .bed binary file, and reads the files into the "unsigned __int64" datatype for efficient storage in the C program. The reading is done using the fread function in C.

The builder.py file uses the C Foreign Function Interface package to make the functions available to the rest of the package, that is written in python.

The bed_reader_dask function then converts the data into the DASK array, it uses the C function to obtain the data, and converts it into a contiguous array of integers per chunk, and then uses the concatenate, delayed and from_delayed functions from the dask package to convert the whole dataset into a DASK array.

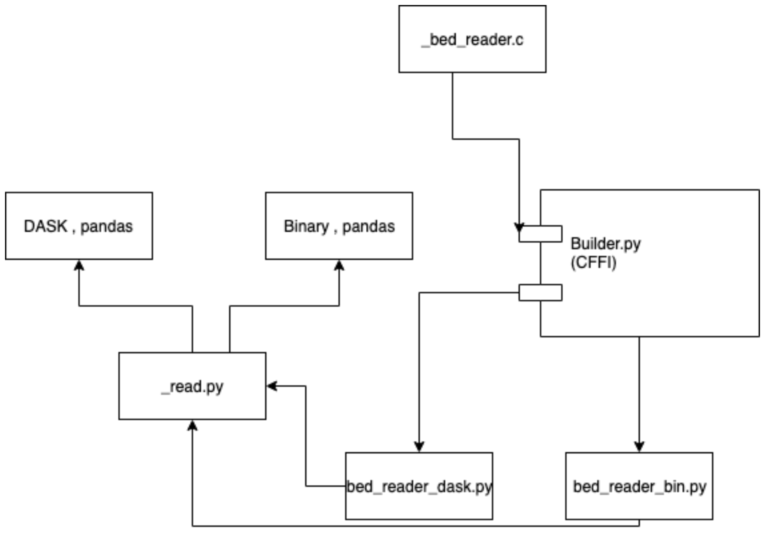


Figure 5: Flow diagram of py_plink package

The bed_reader_bin function converts the data into the binary encoding representation, it uses the BitVector package, and store each SNP as a BitVector. The dataset is therefore converted into a list of BitVectors, where each BitVetor represents a SNP.

The _read.py file performs a check on the bed file data to ensure that it is formatted correctly and converts the .bim and .fam files into pandas dataframes. The read_plink function takes the filename and output format as an input, and outputs the respective bim, bed and fam arrays as a tuple.

## 5 Statistical Hypothesis Tests

Statistical Hypothesis Tests are used to test whether the associations seemingly found when a locus of SNPs is observed for the group of Cases (the hypothesis), is valid, and not just a matter of chance. These tests produce a value of statistical power, which is a measure of how strong the proposed hypothesis is. These tests can be used as an objective function in our program and as a measure of fitness for our models.

### 5.1 Pearson's Chi-Squared Test

Pearson's $\chi^2$ test is a statistical hypothesis test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance. The test statistic calculated is then compared to a $\chi^2$ distribution to obtain a p-value.[29]

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

In the context of a GWAS, it can be used as an objective function that uses the marginal probabilities of the cases and controls, to calculate a value of statistical power.

This can be done by defining a null-hypothesis, that there are no phenotypic associations to the SNP, and defining the alternative hypothesis as the SNP being associated to the data. We then compare the expected values of the case and control data with respect to the SNP that we are considering, to the observed value that we obtain from the GWAS study.

After this we can plug the marginal values into the formula, and we obtain a $\chi^2$ value, which we can use in our fitness function. We can also calculate the p-value, by comparing the $\chi^2$ value to a $\chi^2$ distribution, depending on the degrees of freedom.

## 5.2 Chi-Square Test for Single Interaction

In this section, we will describe the Chi-Squared test for single interaction analysis, that test if a SNP is associated with a phenotypic trait.

Let $L$ be a diallelic locus with alleles A and a. Suppose A is the minor allele. We use 0,1,2 to represent the respective genotypes aa, Aa and AA at $L$. Let the cell probabilities be $p_i$ and $q_i$ for the cases and controls respectively, and the cell counts be $n_i$ and $m_i, i = 0, 1, 2$ respectively. The aim of the association analysis with case-control design is to test 3 genotypes have the same distribution in cases and controls. We can therefore define the null hypothesis as:

$$H_0 : p_i = q_i, i = 0, 1, 2$$

A $\chi^2$ test done on the null-hypothesis described above would be in the form of an 2 degrees of freedom Pearson's Chi-Squared Test.

This test has been implemented in the single_assoc class in the Single Association Test Jupyter notebook, and is used to test individual SNPs in the dataset for association with T2D.

## 5.3 Chi-Square Test for Gene-Gene Association

In this section, we will describe the Chi-squared test for gene-gene interaction , that tests if a combination of two SNPs are associated with a phenotypic trait.

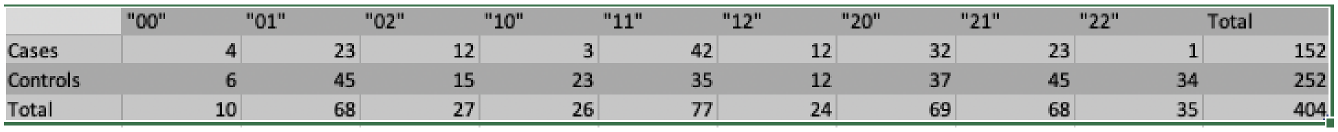
Let $L_1$ and $L_2$ be two unlinked diallelic loci with alleles A, a and B, b, respectively. Suppose A and B are the minor alleles. We use 0, 1, 2 to represent the respective genotypes aa, Aa, AA at $L_1$ and bb, Bb, BB at $L_2$. Let the cell probabilities for cases and controls be $p_{ij}$ , $q_{ij}$ and cell counts be $n_{ij}$ and $m_{ij}$ , i, j = 0, 1, 2, respectively. The aim of association analysis with a case-control design is to test if the 9 genotype combinations have the same distribution in cases and controls. We can therefore define the null hypothesis as:

$$H_0 : p_{ij} = q_{ij}, i, j = 0, 1, 2$$

A $\chi^2$ test done on the null-hypothesis described above would be in the form of an 8 degrees of freedom Pearson's Chi-Squared Test.

This test has been implemented in the aco_gwas class in the Ant Colony Optimisation - GWAS Jupyter Notebook , and is used to test the combined effect of two SNPs in the dataset for association with T2D.

This test is the total chi-squared that measures any violation of the null hypothesis. It will tell us if there is significant interaction, but can not tell us any specifics, if the null hypothesis is rejected. Therefore it is a good test for association, but can not give us any real information about interaction. [30]



|  | "00" | "01" | "02" | "10" | "11" | "12" | "20" | "21" | "22" | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Cases | 4 | 23 | 12 | 3 | 42 | 12 | 32 | 23 | 1 | 152 |
| Controls | 6 | 45 | 15 | 23 | 35 | 12 | 37 | 45 | 34 | 252 |
| Total | 10 | 68 | 27 | 26 | 77 | 24 | 69 | 68 | 35 | 404 |

Table 1: 2x9 Pearson's Chi-Squared Test

### 5.4 Partitioned Chi Squares for Interaction

The $\chi^2$ Test proposed above only tests for association between the two loci, and does not give us any extra information about the allelic interaction factors. We therefore adopt a model that was proposed by Yang et al, in [30] of first, partitioning the $\chi^2$ into a $\chi^2$ for main effects and interaction effects, and then to further partition the test for interaction effects into four separate test for specific groups of combinations, that Yang et al have defined. The first component of the partition, $\chi^2_{I1}$is the chi-square for testing no Aa x Bb interaction effect. Given this is true, $\chi^2_{I2}$ and $\chi^2_{I3}$ are used to test that there are no interaction effects Aa x BB and AA x Bb. The last component, $\chi^2_{I4}$ , is the test for no AA x BB interaction effect given that the previous three interaction effects are absent. The partial sum $\chi^2_{I1} + \chi^2_{I2} + \chi^2_{I3}$ may be used to test that there are no heterozygote effects in interactions (this means that genotype combinations (a a , b b ), (a a , Bb), (Aa , b b ) have the same effects). If this is true, then the last component, $\chi^2_{I4}$ , is the chi-square for interaction under the epistasis model. This model would be an interesting way to find deeper and richer interactions within the algorithms proposed. It would be an interesting direction for the fitness functions of the ACO algorithm to implement in the future.

## 6 Single Association

This section of the report will look at the single association test developed for analysis of the whole dataset. The program is implemented as a python class on a Jupyter notebook. It will be used as a baseline test of our associations found, to see if the interactions are caused by a single main effect, or weather a real epistatic interaction is occurring.

### 6.1 Program Structure

The Single Association Analysis program is implemented as a class, that takes in the bim DASK array, the fam and bed pandas dataframes, and the indexes of the cases and controls as initialisation arguments. After a run, it returns a list of p-values for single association, as well as print out the 20 best SNPs found.

The class uses an allele wise 2x3 $\chi^2$ test with 2 degrees of freedom, to test the statistical significance of each individual SNP in the database. The p-values are then correct using the Bonferroni Correction, which is an adjustment made to P values when several statistical tests are being performed simultaneously on a single data set [31].

The $\chi^2$ calculations are parallelised using the multiprocessing package on python, to maximise computational efficiency and reduce run time.

The program uses the manhattan_plot() function from the post_processing class that is implemented in the ACO notebook . A Manhattan plot is a type of scatter plot, usually used to display data with a large number of data-points, many of non-zero amplitude, and with a distribution of higher-magnitude values [32]. The plot is commonly used in GWAS to display significant SNPs. [33]

### 6.2 Manhattan Plot

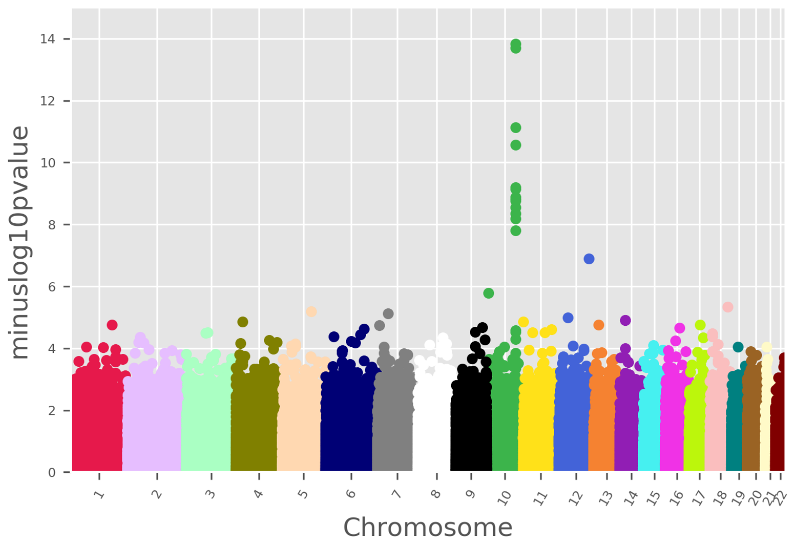


Figure 6: Manhattan Plot of whole dataset

In Figure 6, we can see the individual SNPs corresponded to their $-log_{10}p$-value, this gives us an idea of the location of the significant loci for the phenotypic trait. We can see that the majority of low p-value SNPs are located on chromosome 10.

## 7 Ant Colony Optimisation

Ant-Colony Optimisation is a stochastic optimisation algorithm that is based on the the behaviour of ants in nature. When an ant finds a source of food, it walks back to the colony leaving ”markers” (pheromones) that show the path has food. When other ants come across the markers, they are likely to follow the path with a certain probability. If they do, they then populate the path with their own markers as they bring the food back. As more ants find the path, it gets stronger until there are a couple streams of ants traveling to various food sources near the colony. The pheromone markers fade as time passes.[22]

The ACO algorithm is a strong candidate for the problem, as it has a natural fit with discrete optimisation problems, and with alterations to allow for the selection of subsets of variables and a highly configurable pheromone deposition rule, is well suited to the problem of finding gene-gene interactions in large data. [7] The ACO algorithm incorporates pheromone trails and evaporation but is modified in several ways from a traditional method, with the inclusion of a subset-based pheromone deposition and tournament path selection. The ACO algorithms in this paper have been influenced by the algorithms described by Sapin et al, in [7].

### 7.1 Ant Colony Optimisation for GWAS

The ACO algorithm for detecting gene-gene interactions is run as a stochastic wrapper method. The algorithm selects $N$ SNPs from the full dataset and the combination is evaluated for their ability to differentiate between cases and controls within the dataset.

### 7.2 Algorithm

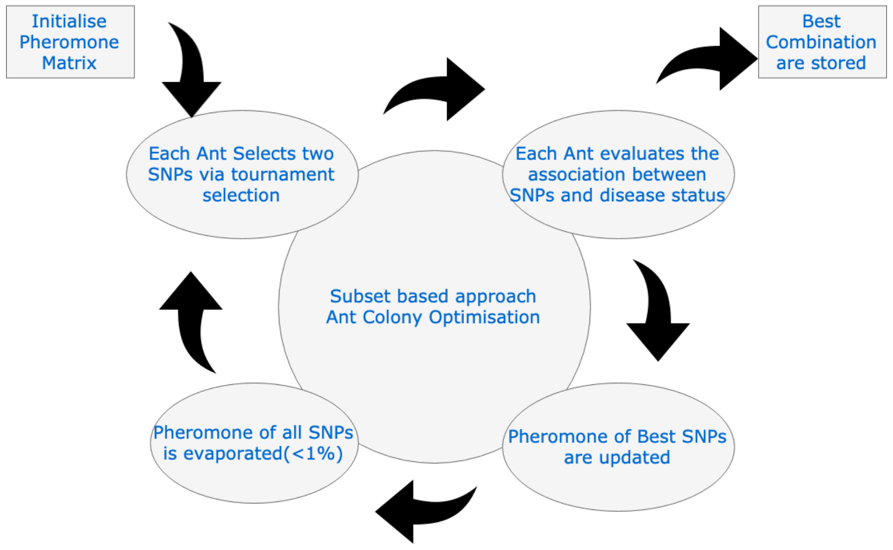

Initialise Pheromone Matrix
Best Combination are stored
Each Ant Selects two SNPs via tournament selection
Each Ant evaluates the association between SNPs and disease status
Subset based approach Ant Colony Optimisation
Pheromone of all SNPs is evaporated(<1%)
Pheromone of Best SNPs are updated

Figure 7: Overview of subset based ACO approach

The Ant Colony Optimisation approach for searching for combinations of SNPs that can discriminate between controls and cases within a GWAS dataset has been described in Figure 7. An ant selects a combination of SNPs from the dataset randomly with a bias towards SNPs with the greatest pheromone value. The fitness of the combination of SNPs to discriminate between the cases and controls is calculated. The pheromone values of the SNPs are then updated on the pheromone matrix. This is repeated for all the ants and then all pheromone values are evaporated by applying a uniform multiplier($< 1$) to the pheromone matrix.

The algorithm can be described as follows:

**Algorithm 1:** Subset Based Ant Colony Optimisation Algorithm

```
Initialise pheromone on each SNP in pheromone matrix to Initpheromone;
for numgen do
    for all nbant ants do
        Select two SNPs via tournament selection;
        Calculate the fitness of the Combination;
    end
    Update pheromone for the two SNPs with best fitness;
    Multiply the pheromone matrix with evaporation_rate;
end
```

Where *nbant* is the number of ants, *initpheromone* is the initial amount of pheromone that is deposited on all the SNPs in the pheromone matrix, and evaporation.

## 7.3 Selection of SNPs

The general path selection strategy of ACO algorithms is to use a form of a biased roulette wheel, that is both stochastic and biased towards the path with the greatest pheromone. This approach will not work for our used case, as the number of possible path selections is extremely high (500,000 SNPs), and it would be extremely computationally expensive to compute. Therefore we make use of tournament selection to achieve stochastic path selection as it has proven to be better for high dimensional problems. [34] In the tournament-based approach, a number of SNPs (nbt) are randomly selected from the possible set to form a tournament and the SNP with the highest amount of pheromone among them is selected as the next destination for the ant. The tournament has many of the same properties as the roulette wheel approach, in that it enables a balance between selecting paths randomly with lower amounts of pheromone and biasing the search towards those with high pheromone.

## 7.4 Fitness Function

The fitness function in this algorithm needs to represent the ability of SNPs to differentiate between cases and controls. Combinations of SNPs with high fitness values will receive more pheromone, and are therefore more likely to be selected for new combinations. The fitness functions that we have experimented with are based on statistical hypothesis tests that are implemented on a binary classification of cases and controls in GWAS.

The fitness function that we used in the ACO approach are the Omnibus Chi Square Test and the Partitioned Chi Square test that have been described in sections 5.3 and 5.4.

The efficiency of the fitness function is of paramount importance to the overall efficiency of the program, it is the main bottleneck of the time taken in execution, therefore any improvement to the fitness function will greatly improve the performance of the overall model.

## 7.5 Updating Pheromone

At each generation of the algorithm, each of the nbant ants selects two SNPs to test their combination. The amount of pheromone of the two SNPs contained in the combination with the highest fitness are updated. For the two pheromone levels the following is applied:

$$P(snp1) = P(snp1) + f(snp1, snp2)$$
$$P(snp2) = P(snp1) + f(snp1, snp2)$$

## 7.6 Parameters

The Ant-Colony Optimisation algorithm is a stochastic search algorithm, and therefore has a set of parameter values that must be set before the experimentation can begin. Standard ACO algorithms usually have the initial value of pheromone deposited, the number of ants, the pheromone evaporation rate and the pheromone deposition as parameters, that will have an effect on the algorithm runs. Our implementation of the ACO doesn't use a standard path selection method, such as biased roulette wheel, and uses tournament selection, therefore the size of the tournament is another parameter that is unique to our ACO implementation. The pheromone deposition was simply the fitness provided by the fitness function that is added onto the existing pheromone.

- nbant: Number of ants in the ACO run
- nbt: Number of SNPs in the tournament
- evaporation rate: Rate by which the pheromone is evaporate after each generation

## 7.7 Program Structure

The implementation of the algorithm is implemented in a Jupyter notebook, using the Python programming language. It involves three main components:

1. The pre_processing class that does the reading in and preprocessing of the data
2. The aco_gwas class that implements an ACO run on the dataset.
3. The post_processing class that analyzes the results of the ACO run.

### 7.7.1 The pre_processing class

The pre_processing class has two functions, not including the initialisation and run functions, called read_pheno() and cases_controls().

The initialisation function takes the following arguments: plink_fn(Plink filename), pheno_fn(Phenotype filename), nbant(Number of Ants), nbt(Size of the tournament), evaporation_rate(evaporation rate), init_val(initial value of pheromone distributed), total_fitness_evals(total number of fitness evaluations) as input. It then uses the py_plink package to convert the .bed file into an array of BitVectors, and converts the bim and fam file to pandas dataframes. It then invokes the read_pheno function to convert the phenotype file into a pandas dataframe, and invokes the cases_controls function to get the indices of the cases and controls.

The read_pheno function is used to read the phenotype text file and convert it to a pandas dataframe. It take fn, which is the filename of the phenotype file, as input and returns df, which is a pandas dataframe of the phenotype file.

The cases_controls function goes through the phenotype dataframe and extracts the indices of the cases and controls, It returns a tuple of cases_i and controls_i, which are the indices of the cases and controls respectively.

The run() function return the list of parameters that are required for the ACO run.

### 7.7.2 aco_gwas class

The aco_gwas class has six functions, not including the initialisation and run functions, called init_pheromone(), tournament_choice(), chi_sq_omnibus(), update_pheromone(), evaporate_pheromone() and ant(). The initialisation function takes the output of the run() function of the pre_processing class as input. It initialises the pheromone_matrix to a NumPy array of length n_snps(the number of SNPs)

The init_pheromone() function initialises the pheromone_matrix to the value initpheromone. The tournament_choice() function performs the tournament selection that was explained in section 7.2, the function return the index of the tournament choice.

The chi_sq_omnibus() takes the indices of two SNPs and then performs the omnibus chi square test that was explained in section 5.3. The function return the chi square statistic, as well as the p-value of the combination.

The update_pheromone() function takes the indices of two SNPs, calls the chi_sq_omnibus() function to obtain their fitness and then updates their fitness according to the pheromone update rule in section 7.5. It also checks the fitness of the combination against the global best fitness and updates it.

The evaporate_pheromone() function multiplies the pheromone_matrix by the evaporation_rate.

The ant() function calls the tournament_selection() and update_pheromone function for a single ant.

The run() function runs the ACO algorithm for numgen genration with nbant ants in each generation. It also records the best fitness after each generation. After a run is complete, the function return the pheromone_matrix, the best fitness for each generation and the overall best p-value of a combination. It also prints out the best combination.

### 7.7.3 post_processing class

The post_processing class has 4 functions, not including the initialisation function, called save_file(), load_file(), best_snps() and manhattan_plot().

The __init()__ function takes a bim and fam array as arguments.

The save_file() function takes a file name and an object to save as arguments. It is used to save the results of an ACO run in a binary file.

The load_file() function takes a filename as input, reads the respective file into an object and then returns the object.

The best_snps() function takes the list of p-values and the number of top SNPs to be printed out as arguments, and prints out the top n_best_snps. The manhattan_plot() function is used to form a Manhattan Plot with the calculated p_values of the SNPs from the models. It takes in the list of p-values and the filename, and saves the plot to a .png file. The function used the matplotlib.pyplopt library to implement the Manhatten Plot.

## 8 Ant Colony Optimisation with Tabu List

### 8.1 Tabu List

A tabu list [35] is a short-term set of the solutions that have been visited in the recent past. The objective for the Tabu Search algorithm is to constrain an embedded heuristic from returning to recently visited areas of the search space, referred to as cycling [35]. The strategy of the approach is to maintain a short term memory of the specific changes of recent moves within the search space and preventing future moves from undoing those changes [35]. A Tabu list can help our ACO algorithm move away from main effects search, by adding the SNPs with the highest fitness to the tabu list.

### 8.2 ACO-Tabu Method

A modification to the standard ACO method, that was explained in the previous section, was proposed by Sapin et al, in [7]. The method proposes using a Tabu list [36] to remove the main effects from the search, and allows the ACO algorithm to find richer combinations, when main effects are removed. This process ensures that the algorithm does not cycle among solutions and can be used to promote promising areas of the search space. The ACO algorithm runs numgen generations, after which the SNP snp1 with the most amount of pheromone is chosen, snp1 is then tested against every other SNP in the dataset for association, and the combination with the highest fitness value is stored, and then snp1 is added to the Tabu list, meaning that it won't get tested again. The ACO-Tabu method can be described by the following algorithm

**Algorithm 2:** ACO-Tabu Algorithm

```
Initialise pheromone on each SNP in pheromone matrix to initpheromone;
for total_fitness_evals do
    Run the ACO algorithm for numgen generations, avoiding the SNPs on the Tabu List;
    Identify the SNP with highest pheromone value as snp1;
    Calculate the fitness of snp1 and all the remaining SNPs;
    Record the best combination ;
    Add snp1 to the Tabu list;
end
```

### 8.3 Program Structure

The ACO-Tabu method uses the pre_processing and post_processing functions form the aco_gwas module, and makes some modifications to the aco_gwas class to add the Tabu list implementation.

The main changes are the addition of two functions, the tabu() and tabu_random() functions, as well as new data members such as tabu_list and best_combination, that are lists that record the tabu list and the best SNP combinations from each run of the SNP that's being added to the Tabu list against all other SNPs.

The tabu() function, implements a tabu run, after numgen generations, adds the best performing SNP to the tabu list, and evaluates its fitness compared to every other SNP in the dataset.

The tabu_random() function generates a list of random number from the range of SNP indices, not including the items in the tabu list.

## 9 Implicit Causal Model

The initial plan that was set out in the Literature Review and Project specification was to experiment with using a Feed Forward Neural Network, as well as a Convalutional Neural Network to detect gene-gene interaction in a Genome-Wide Association Study. After a significant amount of experimentation, and a more in depth look into the prevailing deep learning models for GWAS, it came to my attention that these methods were traditionally applied to reduced sets of SNPs and could not accommodate the size of our dataset. I then came across [37] that proposed a new novel deep learning method based on Bayesian inference and probabilistic modelling, to try to find gene-gene interaction in Genome-Wide Association studies.

Unlike our other GWAS models, that learn statistical relationships, causal models let us manipulate the generative process and make counterfactual statements,i.e., what would happen if the distribution changed. In [37], Tran et al take ideas from causality and modern probabilistic modeling to develop

Implicit Causal Models, a class of causal models that leverages neural architectures with an implicit density to capture important nonlinearities, such as gene-gene interaction.

Implicit models capture an unknown distribution by hypothesizing about its generative process . For a distribution $p(x)$ of observations $x$, the paper defines a function $g$ that takes in noise $\epsilon \sim g(\epsilon|x)$ and outputs $x$ given parameters $\theta$.

$$x = g(\epsilon|\theta), \epsilon \sim s(.)$$

Unlike models which assume additive noise, setting g to be a neural network enables multilayer, nonlinear interactions. The kind of interactions that we are looking for in a model like epistasis. To enforce causality, the paper defines an implicit causal model as a probabilistic causal model where the functions $g$ form structural equations, that is, causal relations among variables

## 9.1 Background: Probabilistic Causal Models

Probabilistic causal models [38] or structural equation models, represent variables as deterministic functions of noise and other variables.

$$\beta = f_\beta(\epsilon_\beta), \epsilon_\beta \sim s(.)$$

For each data point,

$$x_n = f_x(\epsilon_{x,n}, \beta), \epsilon_{x,n} \sim s(.)$$

$$y_n = f_y(\epsilon_{y,n}, x_n, \beta), \epsilon_{y,n} \sim s(.)$$

In this model, all variables are functions of noise $\epsilon \sim s(.)$ and other variables. Each variable $\beta, x, y$ is a function of other variables and its background variable. This allows us to estimate the causal mechanism $f_y$. This allows us to calculate the causal effect $p(y|do(X = x), \beta)$, the probability of an outcome $y$ given that we force $X$ to a specific value $x$ and under fixed global structure $\beta$. The paper infers that due to the causal graph $p(y|do(X = x), \beta) = p(y|x, \beta)$. This means that we can estimate $f_y$ from observational data $\{(x_n, y_n)\}$.

To enforce causality, we define an implicit causal model as a probabilistic causal model where the functions $g$ form structural equations, that is, causal relations among variables.

In [37], Tran et al, suggest that due to the Universal Approximator Theory of Neural Networks [39] that the collection of $g$ functions and some noise source can fine the true causal model of a GWAS. Tran et al, state that Implicit Causal Models are therefore universal approximators of causal models.

## 9.2 Implicit Causal Model Structure

The Implicit Causal Model proposed by Tran et al, consists of two neural networks, the first one to is built from an estimation of a latent confounder $z$, that is related to the overall structure of an individuals genetic profile. The confounder is generated by using a standard normal distribution. The dimensions of the confrounder K is a hyper-parameter in the model. The paper states that the dimension K should be set to the highest value such that the latent space most closely approximates the true population structure but smaller than the total number of SNPs to avoid overfitting.

For GWAS, there are N individuals, each individual has an input vector $M$ of SNPs, $x_n = [x_{n1}, ..., x_{nM}]$ and a scalar outcome $y_n$(the trait). We are looking at how changes to each SNP $X_m$ cause changes to the trait $Y$. Then, the paper defines a confounder represented by a latent variable $z_n$ which influences $x_{nm}$ and $y_n$ for each individual $n$. The model proposed jointly captures $z_n$ and the mechanism for $X_m$

influencing Y. Consider the Implicit Causal Model where for each data point $n = 1, ....., N$ and for each SNP $m = 1, ........, M$

$$z = g_z(\epsilon_{z_n}), \epsilon_{z_n} \sim s(.)$$

$$x_{nm} = g_{x_{nm}}(\epsilon_{x_{nm}}, z_n | w_m), \epsilon_{x_{nm}} \sim s(.)$$

$$y_n = g_y(\epsilon_{y_n}, x_{n,1:M}, z_n | \theta), \epsilon_{y_n} \sim s(.)$$

The function $g_z(.)$ is fixed and each function $g_{x_m}(.|w_m)$ per SNP depends on the confounder and has parameters $w_m$. The function $g_y(, |\theta)$ for the trait depends on the confounder and all SNPs, and it has parameters $\theta$. Priors are placed over the parameters $p(w_m)$ and $p(\theta)$.

## 9.3 Structure

The model on a high level consists of two steps:

1. Generating the matrices for SNPs and Traits
2. Using Inference to train the models

### 9.3.1 Generative process for SNPs

To generate the matrix of SNPs, the model uses an implicit modeling formulation of factor analysis. Using the encoding of SNPs = 0,1,2, a Binomial distribution from 0 to 2 on $x_{nm}$, where n is the number of individuals and m the number of SNPs. The neural network is as follows:

$$logit \pi_{nm} = NN([z_n, w_m] | \phi)$$

The value $w_m$ serve as principal components. The neural network takes an input of dimension 2K and outputs a scalar real value; its weights and biases $\phi$ are shared across SNPs m and individuals n. This enables learning of nonlinear interactions between $z_n$ and $w_m$, preserves the model's conditional independence assumptions, and avoids the complexity of a neural net that outputs the full N x M matrix. We place a standard normal prior over $\phi$.

### 9.3.2 Generative process for traits

To specify the traits, we build on an implicit modeling formulation of linear regression. Formally, for real-valued y $\epsilon$ R, we model each observed trait as

$$y_n = NN([x_{n,1:M}, z_n, \epsilon] | \theta), \epsilon_n \sim Normal(0, 1)$$

The neural net takes an input of dimension M +K +1 and outputs a scalar real value, In the case of our GWAS, 1 or 0.

### 9.3.3 Likelihood Free Variational Inference

The LFVI algorithm on the posterior parameters is used to train the models. At a high level, the algorithm proceeds in two stages. In the first stage, LFVI cycles through the following steps:

- Sample SNP location $m$ and collect the observations at that location from all individuals.
- Use the observations and current estimate of the confounders $z_{1:N}$ to update the $mth$ SNP component $w_m$ and SNP neural network parameters $\phi$.
- Use the observations and current estimate of SNP components $w_{1:M}$ to update the confounders $z_{1:N}$

In the second stage, we infer the posterior of outcome parameters $\theta$ given the inferred confounders from the first stage.

# 10 Experimentation

In this section we will be experimenting with different parameters for our models.

## 10.1 Ant Colony Optimisation

In this section, we will investigate the effect of changing different model parameters that were mentioned in section 7.6. Inspired by [34], we will be running the algorithm with the values 50 and 200 for the number of ants and the values 100, 200, 500, 1000 and 2000 for the tournament size. We will also experiment with 0.99 and 0.95 as the evaporation rates.

### 10.1.1 50 ants

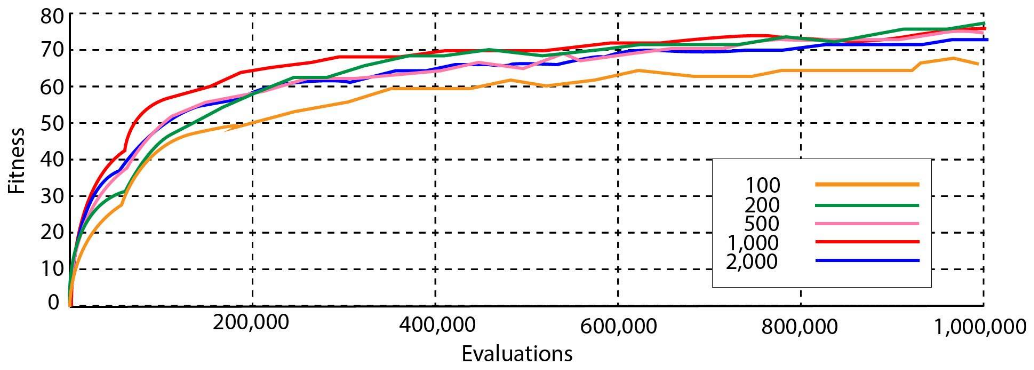


Figure 8: Evolution of the average best fitness over 10 runs of the algorithm with nbant = 50 and nbt = 100, 200, 500, 1,000, 2,000

Figure 8 shows the average highest fitness values that were computed for 50 ants. The best overall fitness was achieved by the 200 tournament size. The 2,000 tournament size shows signs of early convergence and tends to get stuck in local minimas, and the 100,200 and 500 tournament sizes find good fitness values after more than 900,000 fitness evaluation. Therefore, a tournament size of 1,000 has the perfect balance between finding a wider birth of solutions and finding optimal solutions in reasonable time, till 1,000,000 fitness evaluations.

### 10.1.2 200 ants

Figure 9 shows the average highest fitness values that were computed for 50 ants

The best overall fitness was achieved by the tournament size of 2,000 at first and then 1,000 after 700,000 generations and then 500 after a million generations. This shows that the tournament size has a clear effect on the algorithm. It controls its stochasticity against finding good solutions in reasonable time.

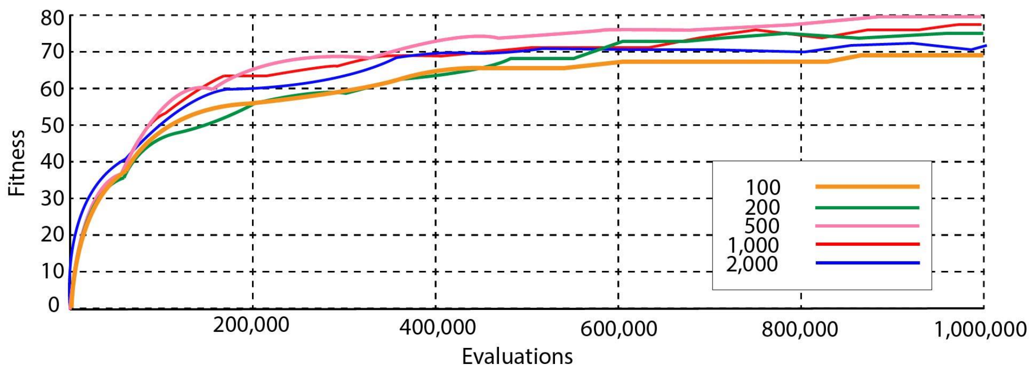
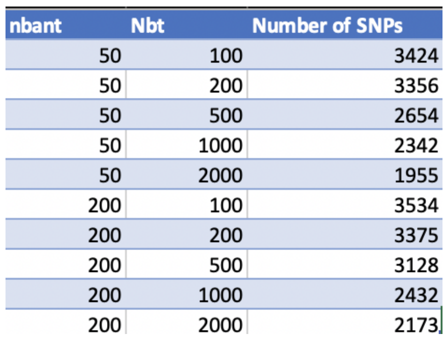


Figure 9: Evolution of the average best fitness over 10 runs of the algorithm with nbant = 200 and nbt = 100, 200, 500, 1,000, 2,000

### 10.1.3 Dataset Coverage

An important factor to consider with the ACO algorithms are their explorative ability within the dataset. The dataset is extremely large( almost 500,000 SNPs) and the ability of the algorithm to explore a reasonable number of solutions in reasonable time. If the search is too restricted, then there is the possibility of missing out a number of key SNPs that could be important for the analysis. In Table 2 we can see the total SNPs that have be accessed in the dataset. Unsurprisingly, the tournament of size 100 explores more of the space than any other, but this might come at the expense of the discovery of better combinations within reasonable time.



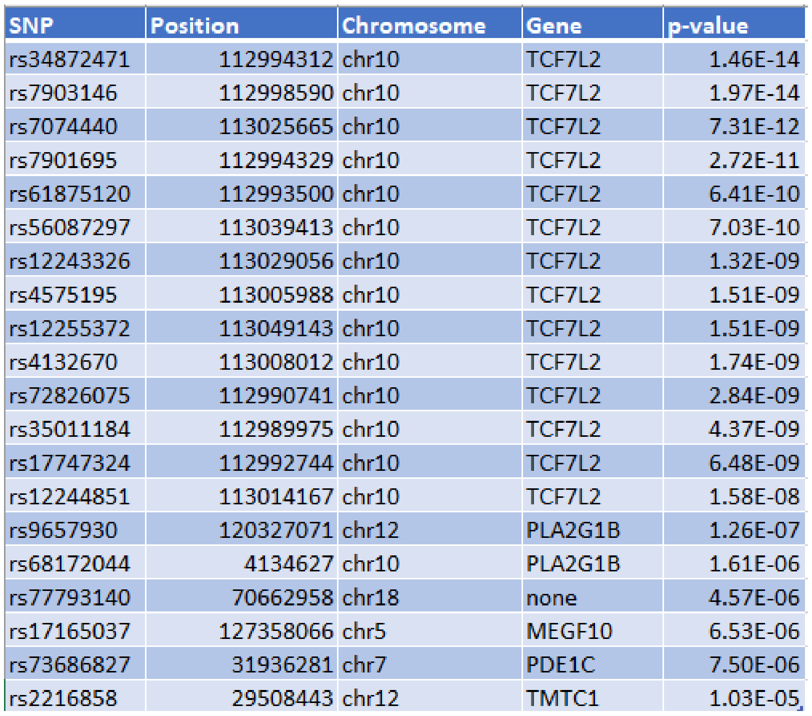
| nbant | Nbt | Number of SNPs |
|---|---|---|
| 50 | 100 | 3424 |
| 50 | 200 | 3356 |
| 50 | 500 | 2654 |
| 50 | 1000 | 2342 |
| 50 | 2000 | 1955 |
| 200 | 100 | 3534 |
| 200 | 200 | 3375 |
| 200 | 500 | 3128 |
| 200 | 1000 | 2432 |
| 200 | 2000 | 2173 |

Table 2: Number of SNPs evaluated for different values of nbant and nbt

## 11 Results

### 11.1 Single Association Analysis

Table 3 showcases the 20 best SNPs found in the Single Association Analysis. The top 16 SNPs, all belong to the TCF7L2 gene sequence. In our multi-locus analyses, we can look for the TCF7L2 gene coming up in an epistatic search as a possible main effect, due to its high single association p-value.



| SNP | Position | Chromosome | Gene | p-value |
|---|---|---|---|---|
| rs34872471 | 112994312 | chr10 | TCF7L2 | 1.46E-14 |
| rs7903146 | 112998590 | chr10 | TCF7L2 | 1.97E-14 |
| rs7074440 | 113025665 | chr10 | TCF7L2 | 7.31E-12 |
| rs7901695 | 112994329 | chr10 | TCF7L2 | 2.72E-11 |
| rs61875120 | 112993500 | chr10 | TCF7L2 | 6.41E-10 |
| rs56087297 | 113039413 | chr10 | TCF7L2 | 7.03E-10 |
| rs12243326 | 113029056 | chr10 | TCF7L2 | 1.32E-09 |
| rs4575195 | 113005988 | chr10 | TCF7L2 | 1.51E-09 |
| rs12255372 | 113049143 | chr10 | TCF7L2 | 1.51E-09 |
| rs4132670 | 113008012 | chr10 | TCF7L2 | 1.74E-09 |
| rs72826075 | 112990741 | chr10 | TCF7L2 | 2.84E-09 |
| rs35011184 | 112989975 | chr10 | TCF7L2 | 4.37E-09 |
| rs17747324 | 112992744 | chr10 | TCF7L2 | 6.48E-09 |
| rs12244851 | 113014167 | chr10 | TCF7L2 | 1.58E-08 |
| rs9657930 | 120327071 | chr12 | PLA2G1B | 1.26E-07 |
| rs68172044 | 4134627 | chr10 | PLA2G1B | 1.61E-06 |
| rs77793140 | 70662958 | chr18 | none | 4.57E-06 |
| rs17165037 | 127358066 | chr5 | MEGF10 | 6.53E-06 |
| rs73686827 | 31936281 | chr7 | PDE1C | 7.50E-06 |
| rs2216858 | 29508443 | chr12 | TMTC1 | 1.03E-05 |

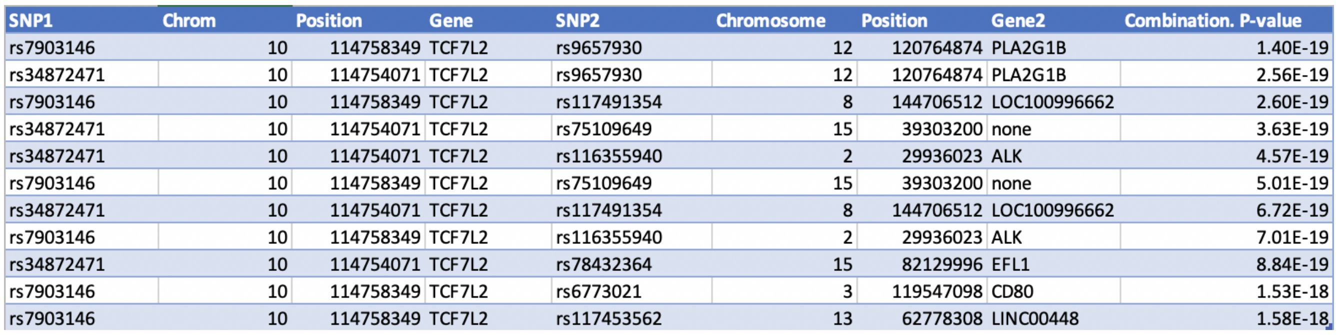
Table 3: 20 Best interactions

### 11.2 Ant-Colony Optimisation

In Table 4, the best results of an ACO run over 1,000,000 fitness evaluations with both 50 and 200 ants as tournament sizes are presented for the Type-II diabetes dataset. The two SNPs that were found in all the best two-locus combinations were the rs7903146 and the rs348724417 SNPs, both of which are in the gene TCF7L2. The TCF7L2 is extensively known to be associated with Type-II diabetes [40]. This demonstrates that the ACO algorithm is able to find SNPs that have been verifiably associated with this disease in the literature. The PLA2G1B gene has also been associated with diet-related obesity and Type-II diabetes in mice [41]. The two best combinations found by the algorithm are both on the TCF7L2 and PLA2G1B genes; this combination could then be investigated further for Type-II diabetes. The SNPs rs117491354(LOC100996662) and rs116355940(ALK) also appeared twice in the analysis and were associated to both the TCF7L2 genes mentioned in the table.

### 11.3 ACO-Tabu Method

In Table 5, the number of times each SNP that has been found at least once over 50 runs of 5,000 generation is shown. A majority of the SNPs added to the Tabu list are from the TCF7L2 gene. The



| SNP1 | Chrom | Position | Gene | SNP2 | Chromosome | Position | Gene2 | Combination. P-value |
|---|---|---|---|---|---|---|---|---|
| rs7903146 | 10 | 114758349 | TCF7L2 | rs9657930 | 12 | 120764874 | PLA2G1B | 1.40E-19 |
| rs34872471 | 10 | 114754071 | TCF7L2 | rs9657930 | 12 | 120764874 | PLA2G1B | 2.56E-19 |
| rs7903146 | 10 | 114758349 | TCF7L2 | rs117491354 | 8 | 144706512 | LOC100996662 | 2.60E-19 |
| rs34872471 | 10 | 114754071 | TCF7L2 | rs75109649 | 15 | 39303200 | none | 3.63E-19 |
| rs34872471 | 10 | 114754071 | TCF7L2 | rs116355940 | 2 | 29936023 | ALK | 4.57E-19 |
| rs7903146 | 10 | 114758349 | TCF7L2 | rs75109649 | 15 | 39303200 | none | 5.01E-19 |
| rs34872471 | 10 | 114754071 | TCF7L2 | rs117491354 | 8 | 144706512 | LOC100996662 | 6.72E-19 |
| rs7903146 | 10 | 114758349 | TCF7L2 | rs116355940 | 2 | 29936023 | ALK | 7.01E-19 |
| rs34872471 | 10 | 114754071 | TCF7L2 | rs78432364 | 15 | 82129996 | EFL1 | 8.84E-19 |
| rs7903146 | 10 | 114758349 | TCF7L2 | rs6773021 | 3 | 119547098 | CD80 | 1.53E-18 |
| rs7903146 | 10 | 114758349 | TCF7L2 | rs117453562 | 13 | 62778308 | LINC00448 | 1.58E-18 |

Table 4: Best two-locus interactions found by the ACO algorithm on the dataset

TCF7L2 gene, a known signal dominates the figure. There are also some other interesting SNPs identified, such as the PLA2G1B gene.

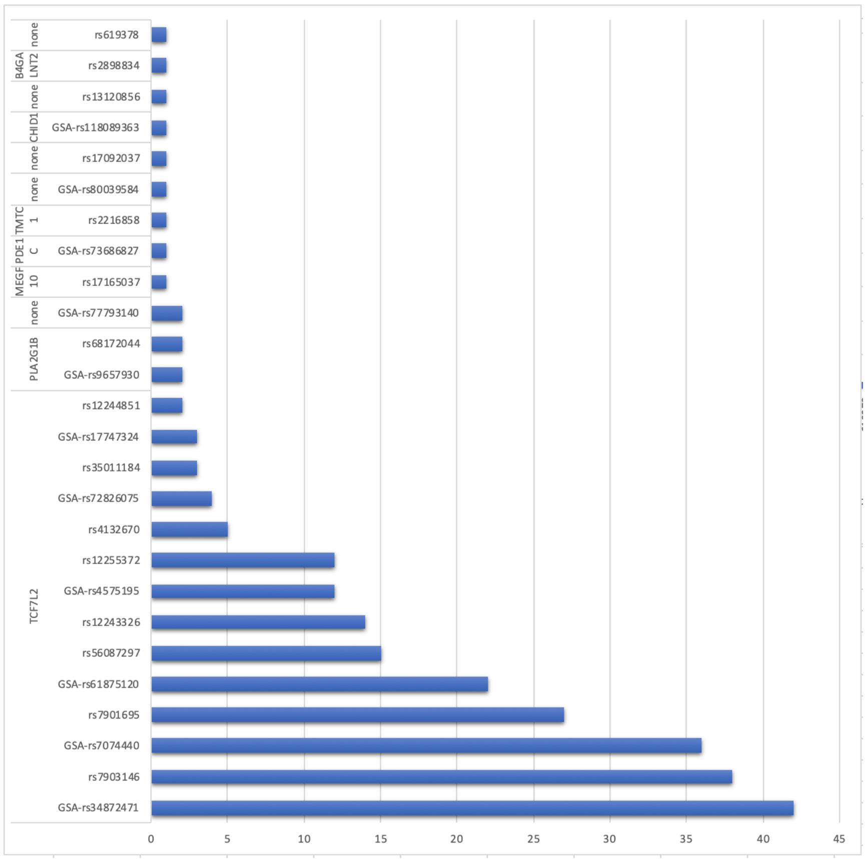


Table 5: Number of times each SNP was found over 50 runs of the ACO-Tabu algorithm and 5,000 generations

# 12 Critical Assessment

This section will contain a Critical Assessment of the project as a whole, highlighting the challenges faced and the difficulties experienced during implementation. This will be followed by suggested changes and possible future directions that the project could be taken in.

## 12.1 Data Representation

One of the main challenges faced with the implementation of the data representation part of the project was the intrinsic slowness of Python as a language. For the binary encoding for example, Decoding the BitVectors back into strings took a considerable amount of time due to the slow standard Python methods, this meant that the implementation of the algorithms was considerably slower. This issue could be solved by moving to a language that was built for speed such as C, or use Cython[42] or PyPy which are Pythonic wrappers on a C style language. This would considerably improve the speed of the data representation.

Another potential direction that the Data Representation section could have taken would be to use the PySnpTools[43] package, which is an end-to-end I/O and representational tool for gnomic data(In the PLINK format), that was developed by Microsoft Genetics Research. This is a quicker, more functional and more memory efficient version of the code that was developed for the data representation.

## 12.2 Statistical Hypothesis Tests

The Statistical Hypothesis Tests were quick and efficient for the most part. The initial implementation of the chi square based fitness function from [7] did not work, the p-values produced were far too large for the dataset, in the degree of 1e-300. This was fixed when the algorithms switched to using the omnibus 2x9 chi square test.

The W-Test[44] is a faster implementation of a statistical hypothesis test on a chi square distribution specifically made for Genome-Wide Association Studies, as well as the partitioned chi square test mentioned in section 5.4, would be good further directions to take this section of the project.

## 12.3 Ant-Colony Optimisation

The ACO algorithms could have also had increased speed if they were developed in a compiled language that would allow for quicker processing speeds, better memory access and quicker runs. The ACO algorithms also need to be permutation tested to establish a threshold of significance for the p-values. The ACO models can be extended to include higher order associations, provided the quicker implementations of the statistical hypothesis tests is implemented.

## 12.4 Implicit Causal Model

hardware limitations made it extremely difficult to implement the Likelihood Free Variational Inference that was required for the training of the Implicit Causal Model, it is for this reason that there are no results for the Implicit Causal Model. The further direction for this model would be to train it on a cluster or on a machine with a powerful GPU. Moreover, I think that the generative model offers a better, more holistic solution to the problem of finding epistatic interactions in a Genome Wide Association Studies, as Neural Networks are universal approximators, and with enough data and appropriate training, should be able to perfectly replicate the causal structure of SNPs that are associated with Type-II Diabetes.

### 12.5 Conclusion

In the introduction, I set out my goals to : ‘Investigate the methodologies and models that have been presented’, ‘Implement these models and test them on the Type-II Diabetes dataset’ and ‘Test their feasibility and efficiency in finding gene-gene interactions’. I believe I have achieved those goals, with the successful implementation of the Ant Colony Optimisation and Ant Colony Optimisation with Tabu List algorithms, that produced excellent results on the Type-II diabetes dataset in reasonable time. Even though the implementation of the Implicit Causal Model was unsuccessful, I believe that I investigated its feasibility thoroughly, and with some better hardware =, could have successfully implemented it. I think the overall combinations found by the algorithms were in accordance with the literature, and some new interesting combinations were also found. I would therefore consider the overall project a success.

## References

[1] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, “Deep learning for computational biology,” *Molecular systems biology*, vol. 12, no. 7, p. 878, 2016.

[2] S. T. Sherry, M. Ward, and K. Sirotkin, “Dbsnp—database for single nucleotide polymorphisms and other classes of minor genetic variation,” *Genome research*, vol. 9, no. 8, pp. 677–679, 1999.

[3] T. A. Manolio, “Genomewide association studies and assessment of the risk of disease,” *New England journal of medicine*, vol. 363, no. 2, pp. 166–176, 2010.

[4] `https://www.genome.gov/20019523/genomewide-association-studies-fact-sheet/`, *Genome-wide association studies*, Accessed:27-08-15.

[5] W. T. C. C. Consortium *et al.*, “Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls,” *Nature*, vol. 447, no. 7145, p. 661, 2007.

[6] G. Davies, R. E. Marioni, D. C. Liewald, W. D. Hill, S. P. Hagenaars, S. E. Harris, S. J. Ritchie, M. Luciano, C. Fawns-Ritchie, D. Lyall, *et al.*, “Genome-wide association study of cognitive functions and educational attainment in uk biobank (n= 112 151),” *Molecular psychiatry*, vol. 21, no. 6, p. 758, 2016.

[7] E. Sapin, E. Keedwell, and T. Frayling, “An ant colony optimization and tabu list approach to the detection of gene-gene interactions in genome-wide association studies [research frontier],” 2015.

[8] T. D. Petes and D. Botstein, “Simple mendelian inheritance of the reiterated ribosomal dna of yeast,” *Proceedings of the National Academy of Sciences*, vol. 74, no. 11, pp. 5091–5095, 1977.

[9] J. H. Moore and B. C. White, “Exploiting expert knowledge in genetic programming for genome-wide genetic analysis,” in *Parallel Problem Solving from Nature-PPSN IX*, Springer, 2006, pp. 969–977.

[10] `https://www.diabetes.co.uk/diabetes-prevalence.html`, *Diabetes uk figures*, Accessed:06-06-18.

[11] T. Nobori, K. Miura, D. J. Wu, A. Lois, K. Takabayashi, and D. A. Carson, “Deletions of the cyclin-dependent kinase-4 inhibitor gene in multiple human cancers,” *Nature*, vol. 368, no. 6473, p. 753, 1994.

[12] `www.ghr.nlm.nih.gov/primer/genomicresearch/snp`, *What are single nucleotide polymorphisms (snps)?* Accessed:07-11-18.

[13] W.-H. Wei, G. Hemani, and C. S. Haley, “Detecting epistasis in human complex traits,” *Nature Reviews Genetics*, vol. 15, no. 11, p. 722, 2014.

[14] H. J. Cordell, “Epistasis: What it means, what it doesn’t mean, and statistical methods to detect it in humans,” *Human molecular genetics*, vol. 11, no. 20, pp. 2463–2468, 2002.

[15] X. Wang, R. C. Elston, and X. Zhu, “The meaning of interaction,” *Human heredity*, vol. 70, no. 4, pp. 269–277, 2010.

[16] B. Mieth, M. Kloft, J. A. Rodrıguez, S. Sonnenburg, R. Vobruba, C. Morcillo-Suárez, X. Farré, U. M. Marigorta, E. Fehr, T. Dickhaus, *et al.*, “Combining multiple hypothesis testing with machine learning increases the statistical power of genome-wide association studies,” *Scientific reports*, vol. 6, p. 36 671, 2016.

[17] H. J. Cordell, “Detecting gene–gene interactions that underlie human diseases,” *Nature Reviews Genetics*, vol. 10, no. 6, p. 392, 2009.

[18] J. H. Moore, F. W. Asselbergs, and S. M. Williams, “Bioinformatics challenges for genome-wide association studies,” *Bioinformatics*, vol. 26, no. 4, pp. 445–455, 2010.

[19] K. L. Lunetta, L. B. Hayward, J. Segal, and P. Van Eerdewegh, “Screening large-scale association study data: Exploiting interactions using random forests,” *BMC genetics*, vol. 5, no. 1, p. 32, 2004.

[20] `https://medium.com/machinevision/overview-of-neural-networks`, *Overwiew of neural networks*, Accessed:07-11-18.

[21] F. Günther, N. Wawro, and K. Bammann, “Neural networks for modeling gene-gene interactions in association studies,” *BMC genetics*, vol. 10, no. 1, p. 87, 2009.

[22] M. Dorigo and M. Birattari, “Ant colony optimization,” in *Encyclopedia of machine learning*, Springer, 2011, pp. 36–39.

[23] X. Wan, C. Yang, Q. Yang, H. Xue, X. Fan, N. L. Tang, and W. Yu, “Boost: A fast approach to detecting gene-gene interactions in genome-wide case-control studies,” *The American Journal of Human Genetics*, vol. 87, no. 3, pp. 325–340, 2010.

[24] `https://exetercrfnihr.org/about/exeter-10000/`, *Exeter 10,000*, Accessed:06-06-18.

[25] `http://microsoftgenomics.github.io/PySnpTools/`, *Microsoft genetics*, Accessed:06-06-18.

[26] J. Christmas, E. Keedwell, T. M. Frayling, and J. R. Perry, “Ant colony optimisation to identify genetic variant association with type 2 diabetes,” *Information Sciences*, vol. 181, no. 9, pp. 1609–1622, 2011.

[27] `http://docs.dask.org/en/latest/array.html`, *Dask array documentation*, Accessed:07-11-18.

[28] `https://towardsdatascience.com/why-every-data-scientist-should-use-dask-81b2b850e15b`, *Why every data scientist should use dask?* Accessed:06-06-18.

[29] M. J. Slakter, “A comparison of the pearson chi-square and kolmogorov goodness-of-fit tests with respect to validity,” *Journal of the American Statistical Association*, vol. 60, no. 311, pp. 854–858, 1965.

[30] Y. Yang, C. He, and J. Ott, “Testing association with interactions by partitioning chi-squares,” *Annals of human genetics*, vol. 73, no. 1, pp. 109–117, 2009.

[31] E. W. Weisstein, “Bonferroni correction,” 2004.

[32] S. D. Turner, “Annotated manhattan plots and qq plots for gwas using r, revisited,” 2011.

[33] G. Gibson, “Hints of hidden heritability in gwas,” *Nature genetics*, vol. 42, no. 7, p. 558, 2010.

[34] E. Sapin and E. Keedwell, “T-aco tournament ant colony optimisation for high-dimensional problems.,” in *IJCCI*, 2012, pp. 81–86.

[35] `http://www.cleveralgorithms.com/nature-inspired/stochastic/tabu_search.html`, *Tabu list*, Accessed:06-06-18.

[36] S. Tsubakitani and J. R. Evans, “Optimizing tabu list size for the traveling salesman problem,” *Computers & Operations Research*, vol. 25, no. 2, pp. 91–97, 1998.

[37] D. Tran and D. M. Blei, “Implicit causal models for genome-wide association studies,” *arXiv preprint arXiv:1710.10742*, 2017.

[38] Y. Peng and J. A. Reggia, “A probabilistic causal model for diagnostic problem solving part i: Integrating symbolic causal inference with numeric probabilistic inference,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 2, pp. 146–162, 1987.

[39] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[40] T. Jin and L. Liu, *The wntsignalig pathway effector tcf7l2 and type 2 diabetes mellitus. molendocrinol 2008; 22 (11): 2383-92.*

[41] E. D. Labonté, P. T. Pfluger, J. G. Cash, D. G. Kuhel, J. C. Roja, D. P. Magness, R. J. Jandacek, M. H. Tschöp, and D. Y. Hui, “Postprandial lysophospholipid suppresses hepatic fatty acid oxidation: The molecular link between group 1b phospholipase a2 and diet-induced obesity,” *The FASEB Journal*, vol. 24, no. 7, pp. 2516–2524, 2010.

[42] `https://cython.org/`, *Cython*, Accessed:06-06-18.

[43] `https://github.com/MicrosoftGenomics/PySnpTools`, *Pysnptools*, Accessed:06-06-18.

[44] M. H. Wang, R. Sun, J. Guo, H. Weng, J. Lee, I. Hu, P. C. Sham, and B. C.-Y. Zee, “A fast and powerful w-test for pairwise epistasis testing,” *Nucleic acids research*, vol. 44, no. 12, e115–e115, 2016.