Topic III: Parsimony analysis for population genetics

Han Yong Wunrow
CSCI 5481

5/12/2017

Contents

1 Introduction **1**
- 1.1 Problems . . . 1
- 1.2 Data . . . 1

2 Implementation **2**
- 2.1 Problem 1: Neighbor Joining Algorithm . . . 2
- 2.2 Problem 2: Sankoff Algorithm . . . 2
- 2.3 Problem 3: Tree Search Algorithm . . . 3

3 Results **3**
- 3.1 Problem 1: Neighbor Joining Algorithm . . . 3
- 3.2 Problem 2: Sankoff Algorithm . . . 5
- 3.3 Problem 3: Tree Search Algorithm . . . 6
  - 3.3.1 Sub Sub Population . . . 6
  - 3.3.2 Sub Population . . . 6
  - 3.3.3 Full Population . . . 9

4 Discussion **11**
- 4.1 Nearest-Neighbor Interchange Algorithm Analysis . . . 11
  - 4.1.1 Pros . . . 11
  - 4.1.2 Cons . . . 11
- 4.2 Final Tree . . . 11

1 Introduction

In this project, we implement a tree search algorithm for parsimony analysis and apply the algorithm to analyze population genetics data. There are three main tree search algorithms mentioned in the text: nearest-neighborhood interchange (NNI), subtree pruning and regrafting (SPR), and the tree bisection and reconnection (TBH) methods. Each of these algorithms make changes to the topology of a generated phylogentic tree and it is then checked to see if has a lower parsimony score according to the Sankoff Algorithm. This will in turn optimize the representation of evolution of our populations. In this project, we will use NNI as our tree search algorithm.

1.1 Problems

1. Use all the SNPs on Mitochondrial DNA and apply the neighbor-joining algorithm to construct a phylogenetic tree of all the individuals.

2. Apply your Sankoff algorithm implemented in HW4 on the phylogenetic tree for parsimony analysis of all the SNPs on Mitochondrial DNA .

3. Implement some tree search strategies discussed in Chapter 8 and perform the same parsimony analysis. Compare your results under different trees in the study. Note that your search algorithm should consider there are families in the population data. Think about how to revise your tree branch interchange algorithm.

1.2 Data

SNP data of Mitochondrial DNA in the HAPMAP project from here. The Hapmap file format is a table which consists of 11 columns plus one column for each sample genotyped. The first row contains the header labels of your samples, and each additional row contains all the information associated with a single SNP. The fields are as follows:

- **rs#** contains the SNP identifier;
- **alleles** contains SNP alleles according to NCBI database dbSNP;
- **chrom** contains the chromosome that the SNP was mapped;
- **pos** contains the respective position of this SNP on chromosome;
- **strand** contains the orientation of the SNP in the DNA strand. Thus, SNPs could be in the forward (+) or in the reverse (-) orientation relative to the reference genome;
- **assembly#** contains the version of reference sequence assembly (from NCBI);
- **center** contains the name of genotyping center that produced the genotypes;
- **protLSID** contains the identifier for HapMap protocol;
- **assayLSID** contain the identifier HapMap assay used for genotyping;
- **panelLSID** contains the identifier for panel of individuals genotyped;
- **QCcode** contains the quality control for all entries;
- subsequently, the list of sample names.

Our dataset consists of Mitochondrial SNP data from 11 global ancestry groups. The IDs for each group are as follows:

ID	Population
ASW	African Caribbean in Barbados
CEU	Utah residents with Northern and Western European ancestry
CHB	Han Chinese in Bejing, China
CHD	Chinese in metropolitan Denver, CO
GIH	Gujarati Indian in Houston,TX
JPT	Japanese in Tokyo, Japan
LWK	Luhya in Webuye, Kenya
MEX	Mexican ancestry in Los Angeles, California
MKK	Maasai in Kinyawa, Kenya
TSI	Toscani in Italia
YRI	Yoruba in Ibadan, Nigeria

Table 1: HAPMAP IDs

2 Implementation

2.1 Problem 1: Neighbor Joining Algorithm

To create our phylogenetic tree we first create a sequence for each individual composed only by the SNPs. Assuming that the individuals will have the same nucleotide for the other positions, the phylogenetic tree will not change if we include the rest of the sequence. We remove duplicate sequences for a better visualization of our tree. Next we align our sequences using the Matlab function multialign using the nuc44 scoring matrix. We then calculate the pairwise distances between sequences using the Matlab function seqpdist using the Jukes-Cantor method and nuc44 scoring matrix. Then with the pairwise distances we construct phylogenetic tree using the Matlab function seqneighjoin.

2.2 Problem 2: Sankoff Algorithm

We utilize our implementation of the Sankoff Algorithm in Homework 3 to compute the parsimony score of our phylogenetic tree. The scoring matrix for Homework3 was used, but with 'T' swapped with 'U'. We wrote two functions to implement this algorithm -

	A	U	T	C	-
A	0	3	4	9	8
U	3	0	2	4	8
G	4	2	0	4	8
C	9	4	4	0	8
-	8	8	8	8	8

Table 2: Scoring Matrix for Sankoff Algorithm

sankoff_main.m and sankoff.m. sankoff.m takes two nodes of phylogenetic tree and computes the minimum parsimony score of the left node plus the score at the right node. sankoff_main then calculates the parsimony score of a phylogenetic tree by iteratively calling sankoff.m.

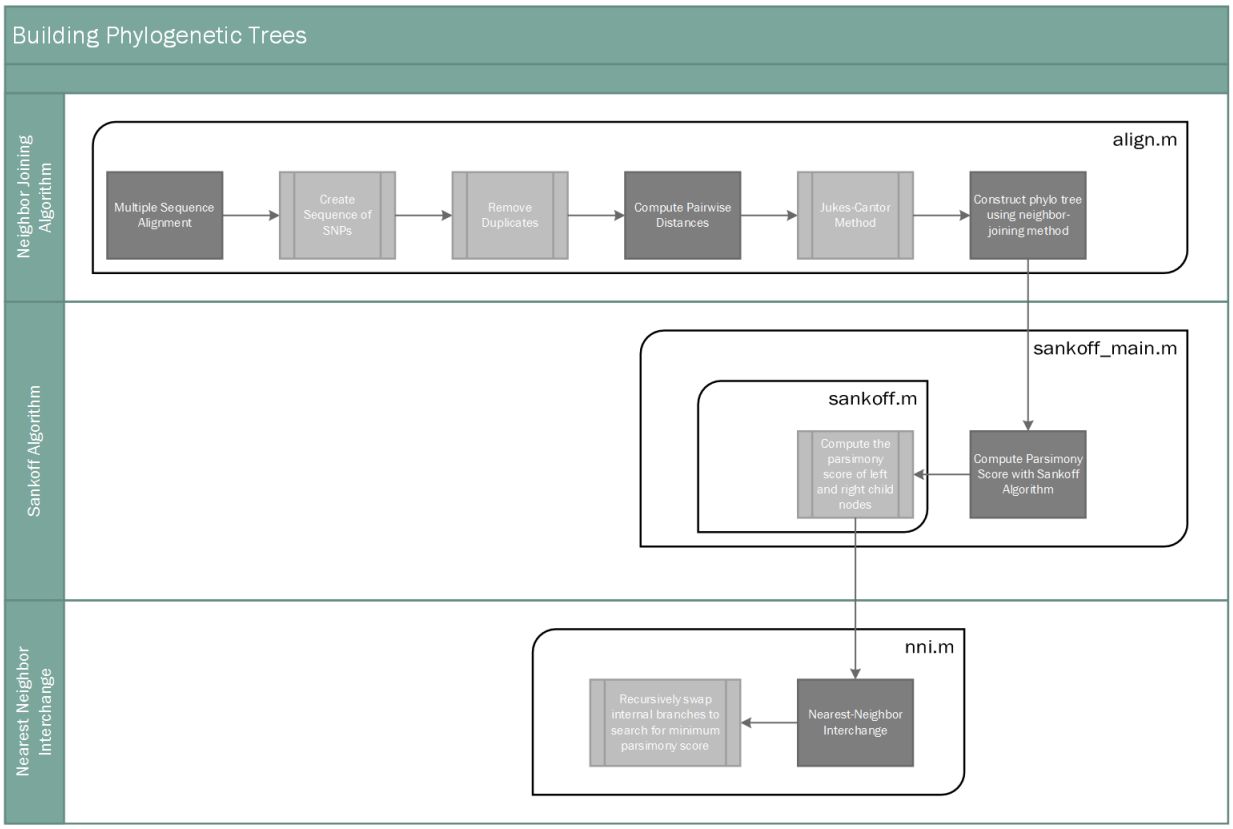

Figure 1: Flowchart of our implementation with each step contained in the corresponding Matlab files

2.3 Problem 3: Tree Search Algorithm

Lastly, we search for the optimal phylogenetic tree by minimizing the parsimony score. There are three main tree search algorithms mentioned in the text: nearest-neighborhood interchange (NNI), subtree pruning and regrafting (SPR), and the tree bisection and reconnection (TBH) methods. NNI takes an internal branches and creates three (two new) possible tree topologies by swapping the four subtrees. Note that one of the three swappings is the original tree. We then check to see if either of the two new tree topologies have a lower parsimony score. NNI then searches recursively for each possible set of subtrees within the tree, making this a slow and rather inefficient strategy. For this project, we focused solely on the implementation and analysis of the NNI strategy.

3 Results

3.1 Problem 1: Neighbor Joining Algorithm

The tree below was produced after running the neighbor-joining algorithm on all 402 sequences. Note that there were 1184 sequences before duplicates were removed. Due to the how large this dataset is, proper analysis of this first phylogeneitc tree is difficult. To get

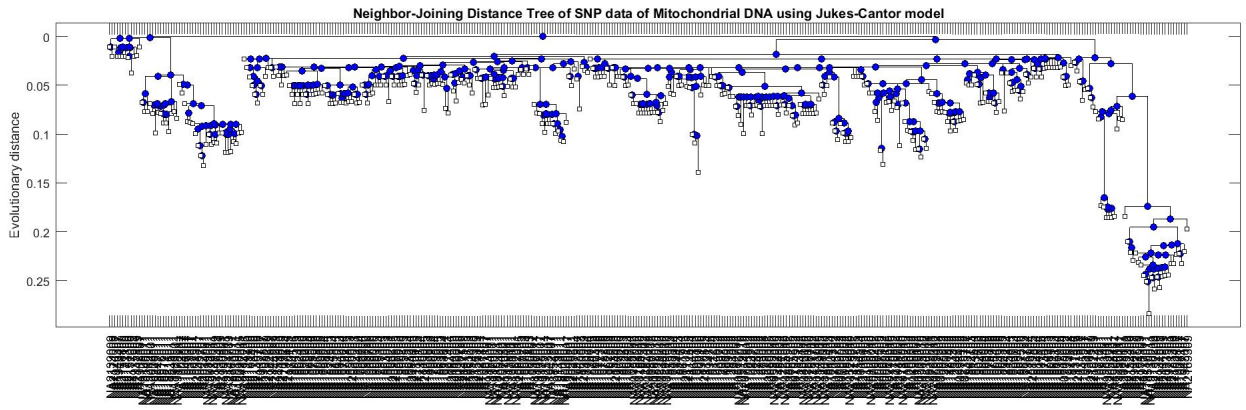

Figure 2: Tree 1

a better visual sense of this tree, we color code each of the leaves by population. We expect that each population (color) should be clustered underneath the same branch. It is

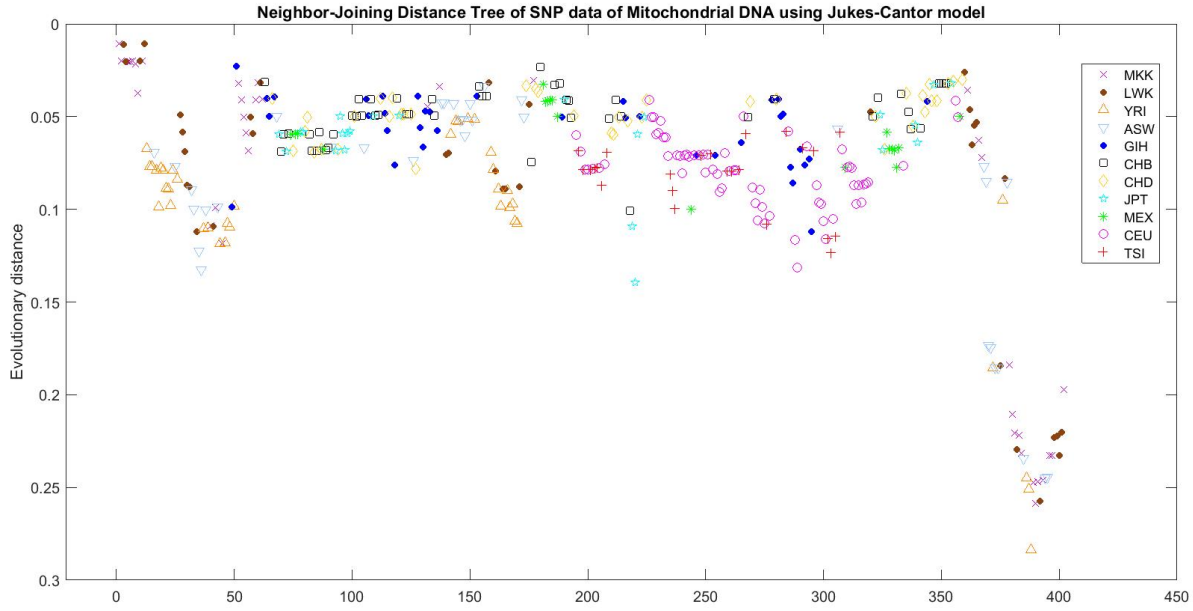

Figure 3: Scatter Plot 1 corresponding to Tree 1

clear in Figure 3 that while the majority of individuals are clustered underneath the same branch, there are a few outliers, especially those from African populations (MKK, LWK, YRI, ASW). Because of this we also include the results of an analysis without these four populations (Figure 4 and 5). It also makes the results of the NNI algorithm more clear as discussed in Section 3.3.

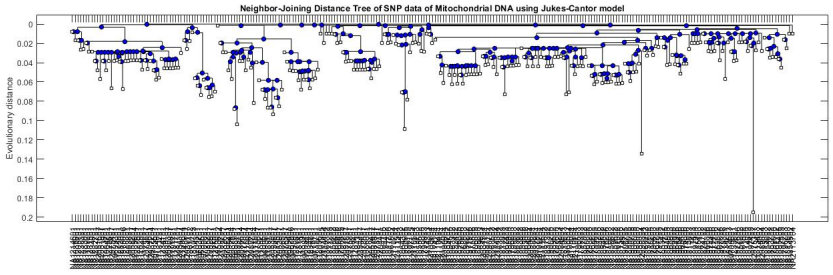

Figure 4: Tree Subpop 1

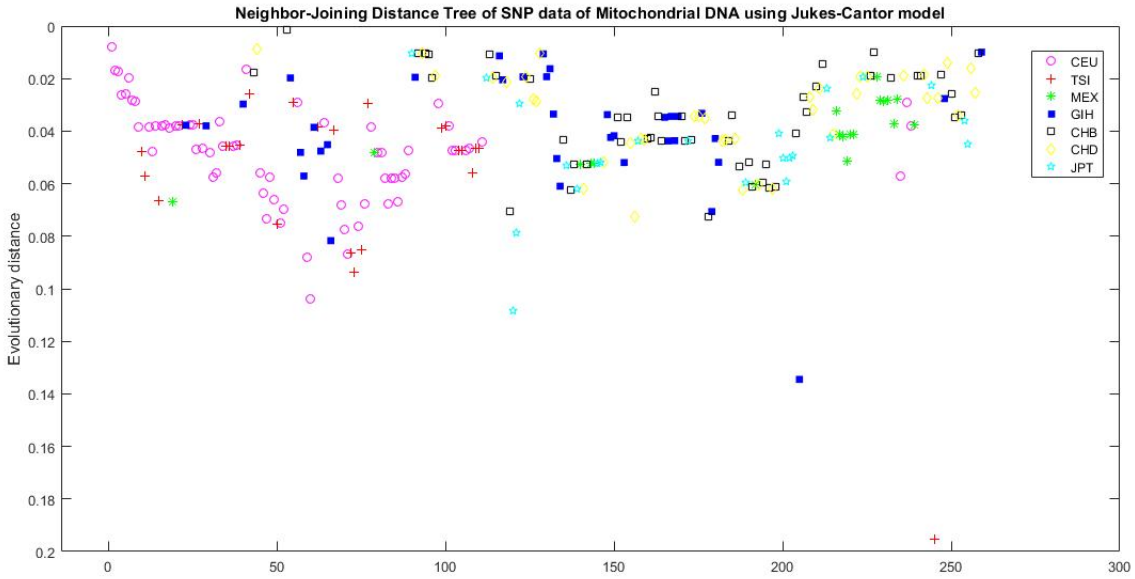

Figure 5: Scatter Plot 1 corresponding to Tree Subpop 1

3.2 Problem 2: Sankoff Algorithm

The parsimony score for Tree 1 in Figure 2 is

$$\text{parsimony(Tree 1)} = 12884$$

The parsimony score for Tree 1 in Figure 4 is

$$\text{parsimony(Tree Subpop 1)} = 7216$$

Both of these scores are fairly arbitrary at this point, but will have more meaning once we compare them to trees created with our tree search algorithms. We will discuss the results of the NNI algorithm in the next section.

3.3 Problem 3: Tree Search Algorithm

3.3.1 Sub Sub Population

For the purpose of visualizing the functionality of our algorithm, we condense our sample to a sub-sub population chosen from Tree Subpop 1 as shown in Figures 6. It is clear to see in the first 3 iterations in Figures 6, 7, and 8 how we swap subtrees as defined by NNI. In this particular case, a tree with lower parsimony score was not found, but trees with lower parsimony scores were found for the subpopulation and full population.

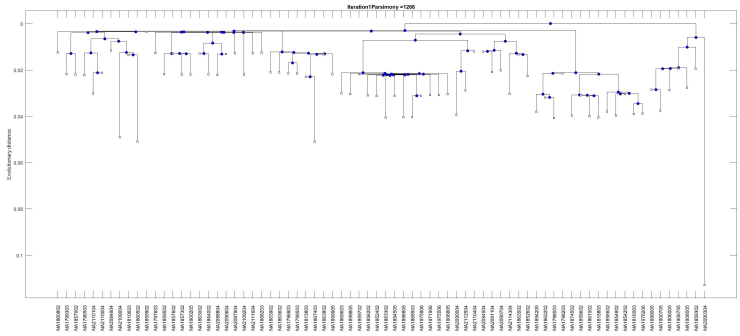

Figure 6: iteration 1

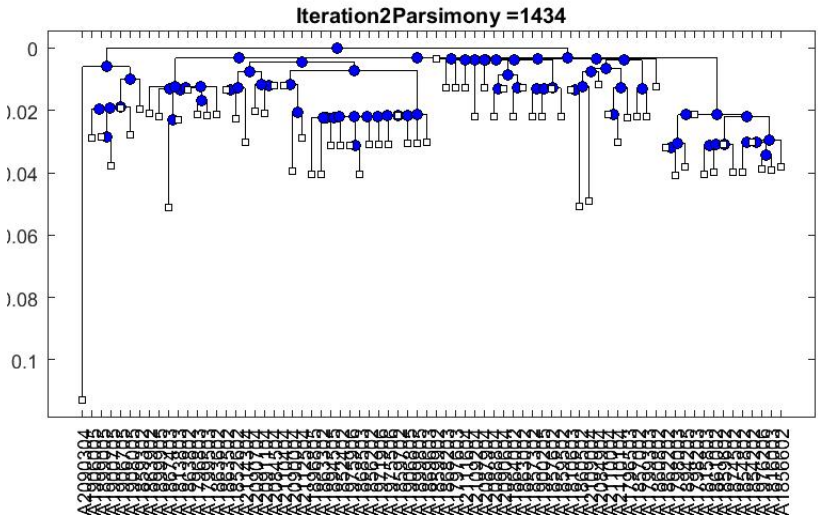

Figure 7: iteration 2

3.3.2 Sub Population

The results after running NNI on Tree Subpop 1 can be seen in Figures 9 and 10. The resulting parsimony score was

$$\text{parsimony(Tree Subpop 2)} = 6668.$$

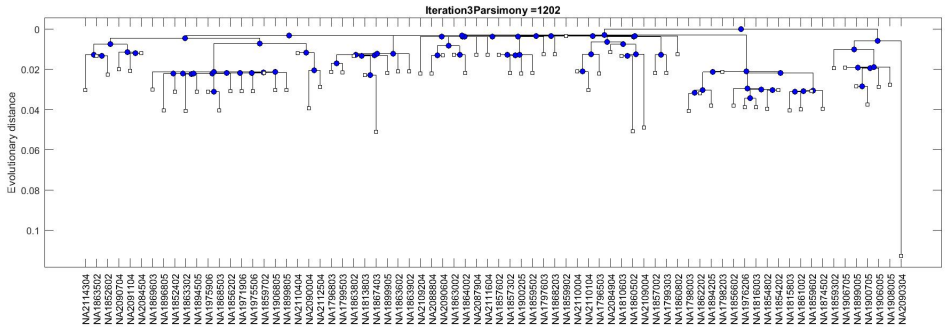

Figure 8: iteration 3

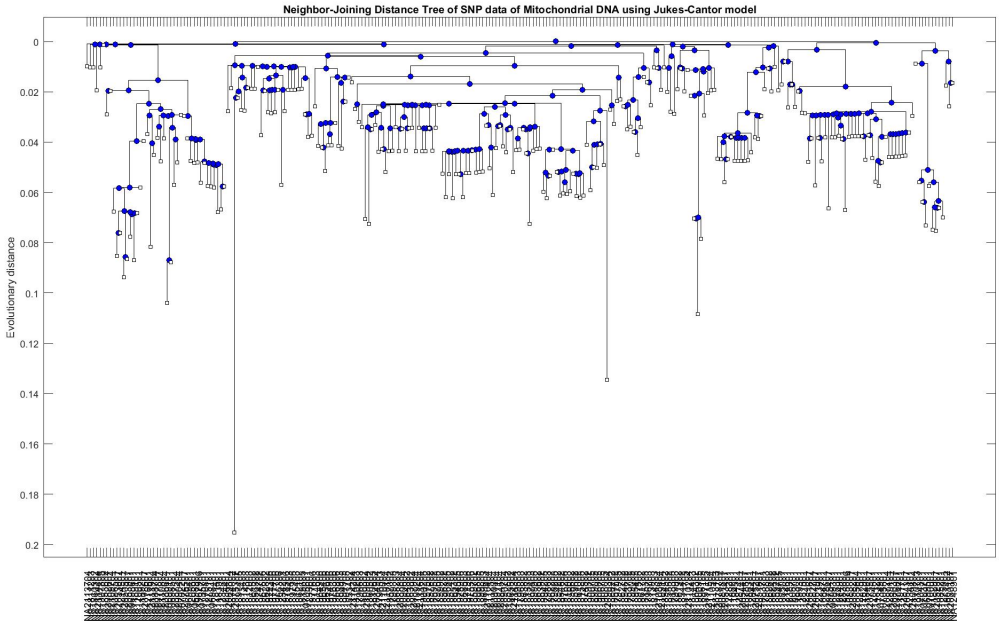

Figure 9: Tree Subpop 2

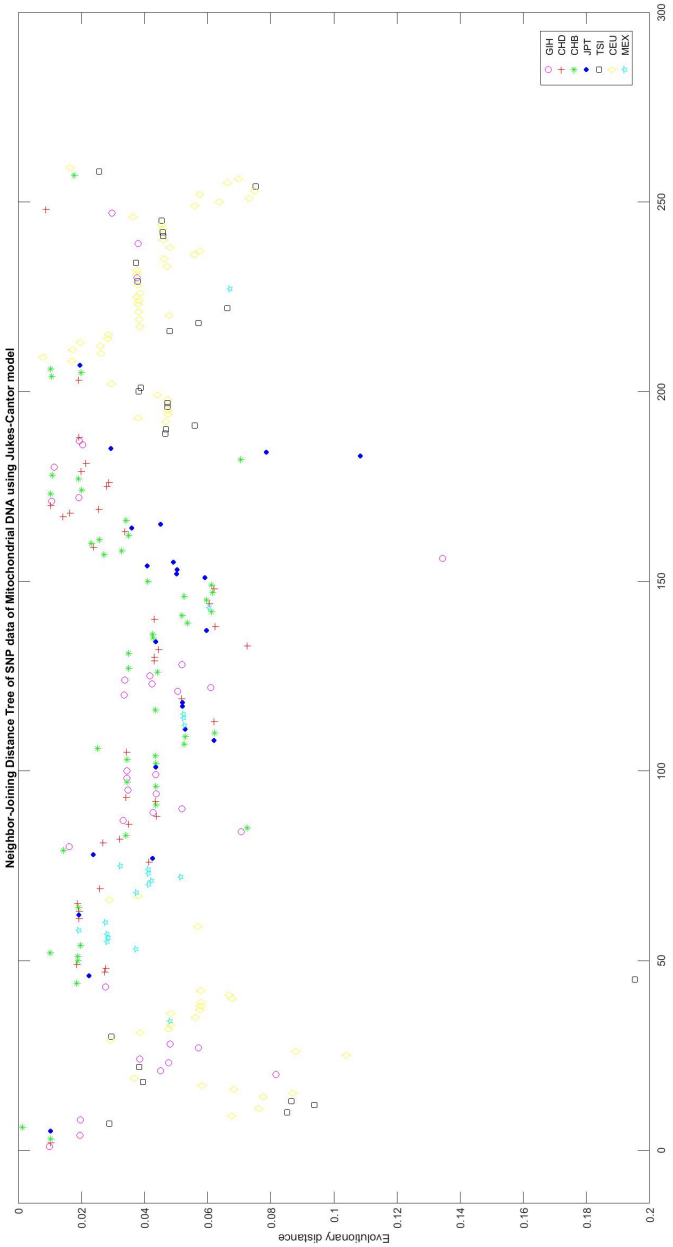

Figure 10: Scatter Plot 1 corresponding to Tree Subpop 2

3.3.3 Full Population

The results after running NNI on Tree 1 can be seen in Figures 11 and 12. The resulting parsimony score was

$$\text{parsimony(Tree 2)} = 11070.$$

This is lower than our original parsimony score by 1814 points. We will discuss the pros and cons of the NNI algorithm and the final result show in Figure 12 in the next section.

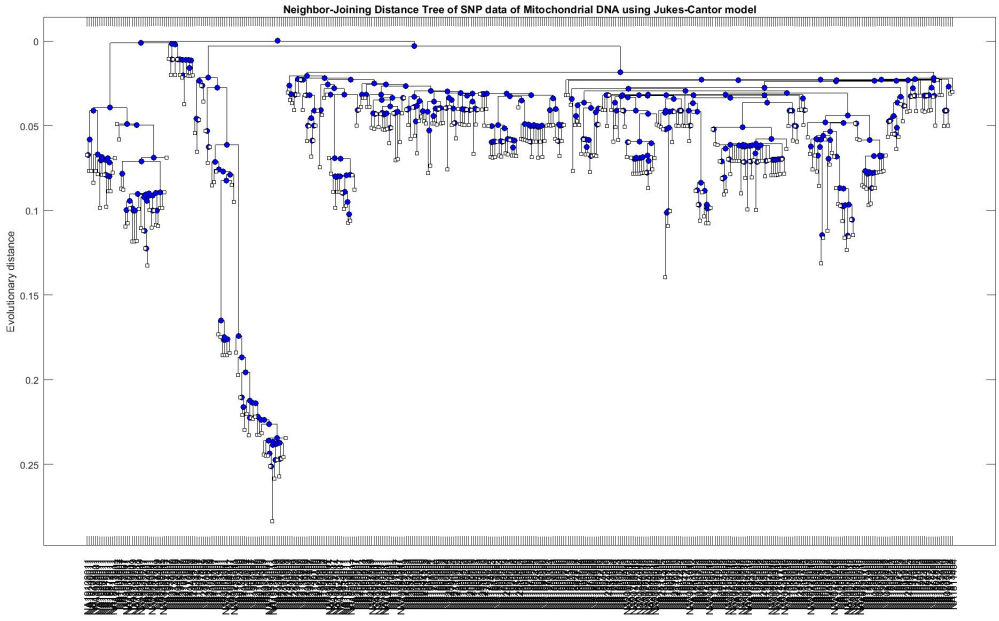

Figure 11: Tree 2

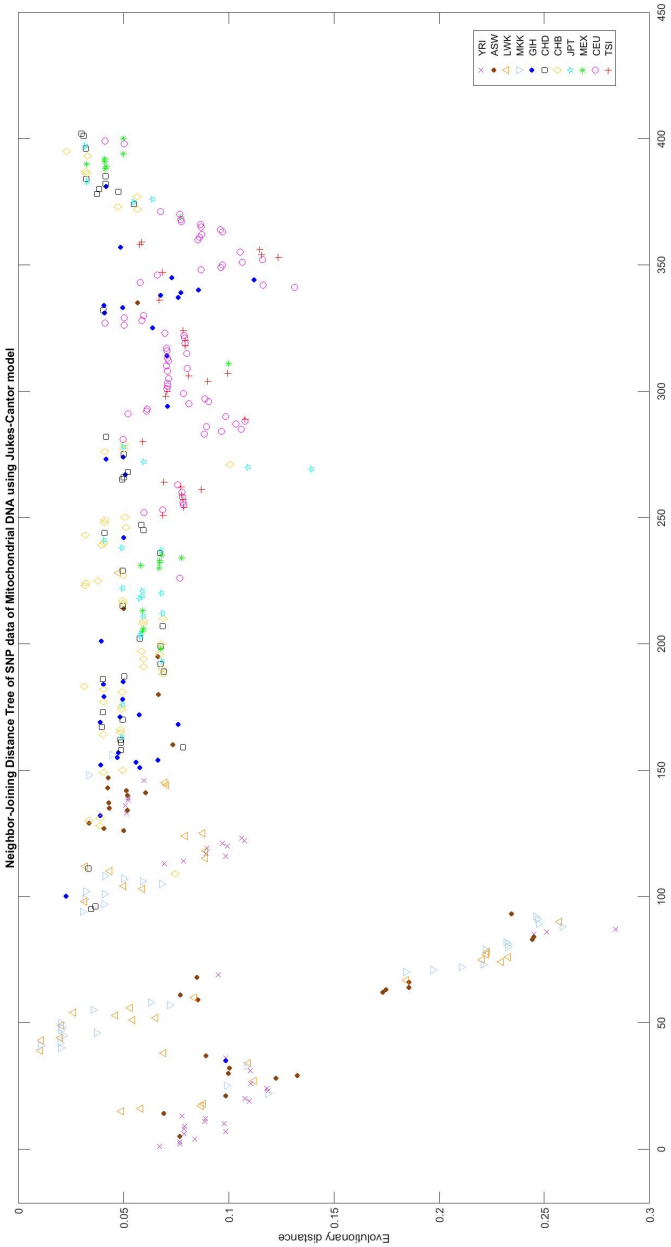

Figure 12: Scatter Plot 1 corresponding to Tree 2

4 Discussion

In this section, we will briefly discuss the pros and cons of the nearest-neighbor interchange tree search strategy and the evolutionary topology the final result on the entire population.

4.1 Nearest-Neighbor Interchange Algorithm Analysis

4.1.1 Pros

Overall, NNI is a effective strategy for enhancing a phylogenetic tree. Since it exhaustively searches the possible nearest-neighbors for each possible set of subtrees it is the slowest but most optimizing way of performing this search. Since we were able to remove duplicate sequences, the runtime of our algorithm was within reason, but for larger datasets it would not work well. With n sequences, there are $2(n-3)$ different trees that this algorithm compares. With large n it is likely that this algorithm will find a more optimal tree with a lower parsimony score, thus making it a useful tree search strategy.

4.1.2 Cons

The main con of NNI is that it is susceptible to local extrema and may not find the global extrema tree topology. Since we can only make small incremental changes to the tree at each step, we are confined to a certain set of trees that are generally very similar to the original tree. SPR and TBR can create larger modifications, and so may be better at finding the global extrema.

4.2 Final Tree

When comparing the original tree in Figure 3 with the final tree in Figure 12 you should note the differences in the colors and symbols for a few of the populations. The improvement to the tree was by moving the 2nd to the left branch from the root to the furthest right position. This moves a number of individuals from African populations to a subtree connected with a number of European and American populations. Although this may seem contadictory, our results are still consistent with the out of Africa Hypothesis, since around 38,500 years ago there a group of African populations (including Yoruba) that branched off with other European, Asian, and American populations as shown on slide 11 of the Phylogeny slides. Further analysis is still needed (for example, implementing in additional tree search algorithms), but SNP data from Mitochondrial DNA serves as a sufficient dataset in creeating phylogenetic trees that are consistent with current ones today! Some further steps include obtaining the SNP data from additional populations as well as implementing SPR and TBR to create additional trees that deviate more from the original.