Nearest Neighbor Joint Algorithm and Its Variations On SNPs of Mitochondrial DNA

Omar Masri
Masri013
5571773
December 20th, 2021

Introduction:

This report paper is a response to the course project for the class CSCI5481: Computational Techniques for Genomics. This project focuses on the creation of the Nearest-Neighbor Algorithm (NNJ), Subtree pruning and regrafting (SPR) which then runs the Sankoff Algorithm to find the parsimony scores to test for efficiency.

I. What is the Nearest-Neighbor Algorithm?
   - The nearest neighbor algorithm is an algorithm that is used to find the most optimal path along a graph of nodes. This is done by repeatedly visiting nearby nodes until an efficient path is found.
   - This can prove to be a little inefficient due to the time complexity necessary to find optimal path
   - Within the realm of DNA sequencing, NNJ is found the algorithms that create and optimize phylogeny trees

II. What is Subtree pruning and grafting?
   - This is an algorithm that chooses a node on a tree returned by the NNJ algorithm. The node is then removed from its current position and reinserted in a different position.
   - The purpose of this algorithm is to check for more optimal parsimony scores that could have been overlooked

III. What is Sank-Off?
   - The Sank-Off algorithm is a scoring algorithm that handles mismatches and gaps
   - This is the algorithm used to find optimal scoring and testing optimal algorithms

IV. Quick Note: I converted all SNP files into Fasta files manually. They all exist in the input file.

Purpose:

In this project the purpose is to test the Nearest-Neighbor algorithm against an algorithm used for what is deemed better optimization. This will be tested through ten different SNPs of Mitochondrial DNA.

Hypothesis:

I believe that throughout my experimentation I will find that the Nearest-Neighbor Algorithm, though slower at giving a result, will be more consistent with its scores. I believe that the SPR algorithm is efficient but it is too random to be used hand in hand with the Sank Off algorithm.

Procedures/Experimentation:

The files provided run back and forth between two main files. Sankoff.py and phylogeny.py.

Sankoff.py:

- Sankoff.py consists of nine functions incharge of deconstructing the sequences and reconstructing them

- It will keep track of the arrays created by each nucleotide which eventually is how the parismory score is found by the algorithm through choosing the minimum number of each array until the root is found.
- The method in which sankoff is implemented by my algorithm is that the sankoff logic will be run on each nucleotide until a root is found, eventually constructing the next sequence, which repeats the process until only one sequence is found.
- The new sequences are the only sequences returned to the files at each step. Meaning the first sequence in the file is the root

Phylogeny.py:

- Phylogeny.py consists of six functions that are in charge of also deconstructing the sequences and reconstructing them.
- This file will return a graphic image of the phylogenetic tree, the selected node to delete and reinsert and then the new tree once the node is reinserted
- Phylogeny's purpose is to create fasta files in the order of the new constructed trees
- Notes:
    - This file will take in fasta files and produce fasta files for the sankoff algorithm. It can be a little confusing in the source code, but all that happens is once the tree and matrix are constructed through the files, they are reconstructed into the fasta files in the order of the new trees and returned to sankoff to find the score.
    - You will have to exit out of each graphic to continue running the program.

Results:

The program is run on ten different SNPs to test the efficiency of both algorithms. The following pages are the results of my experimentation. (This will include pictures from the first two experiments and then will list the results without graphics, you may run the source code to get the graphics. Keep in mind, it will yield different results for the SPR due to its random nature).

The order in which the output file results are formatted are as seen in the following list below. If you wish to ruh the source code, keep in mind which output file you are writing to.

1. genotypes_chrM_ASW_phase3.2_consensus.b36_fwd.fasta

Figure1.a: NNJ

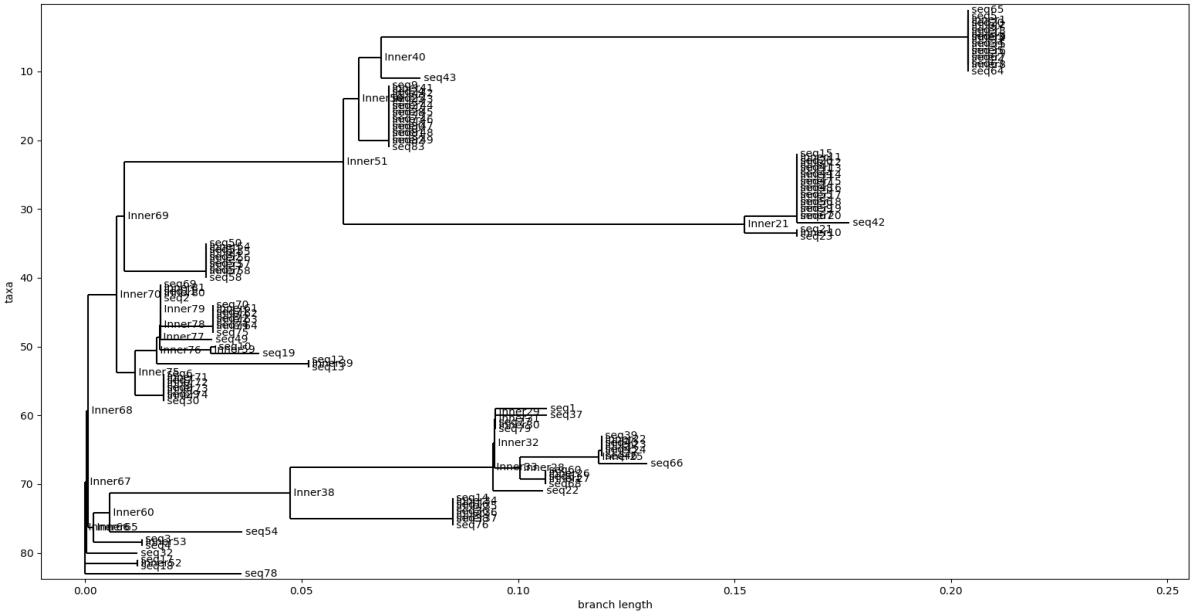
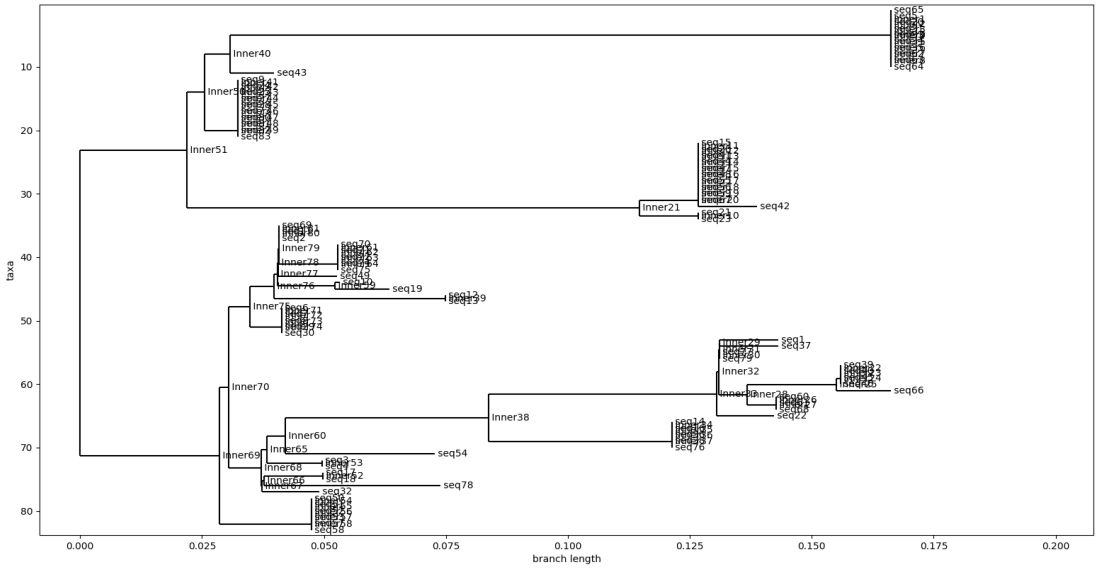


Figure1.b:SPR



NNJ score = 62166
SPR score = 124244

2. genotypes_chrM_CEU_phase3.2_consensus.b36_fwd.fasta

Figure 2.a : NNJ

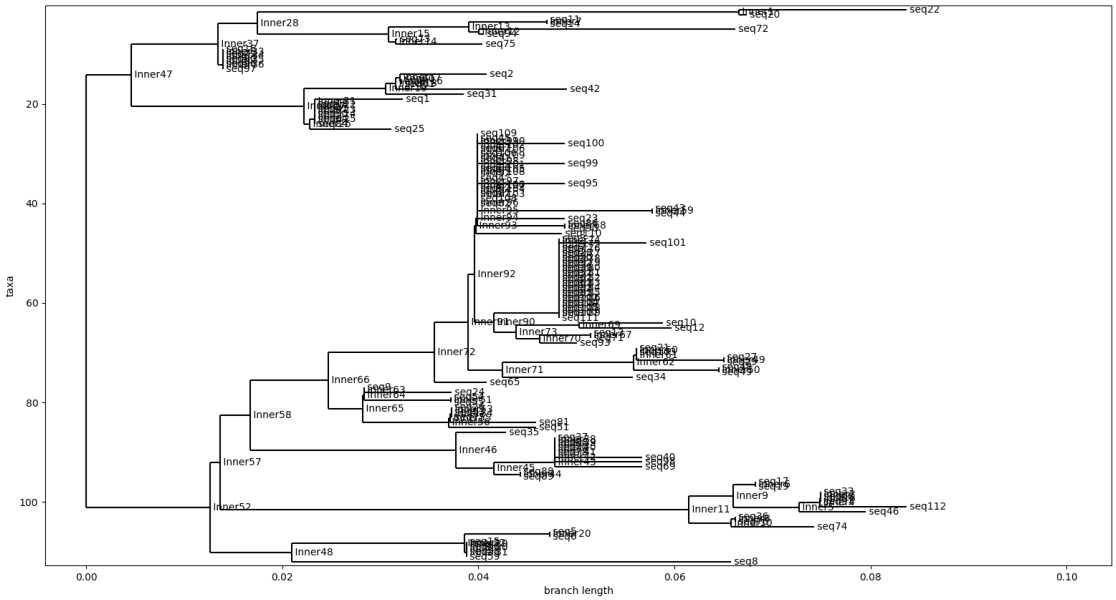
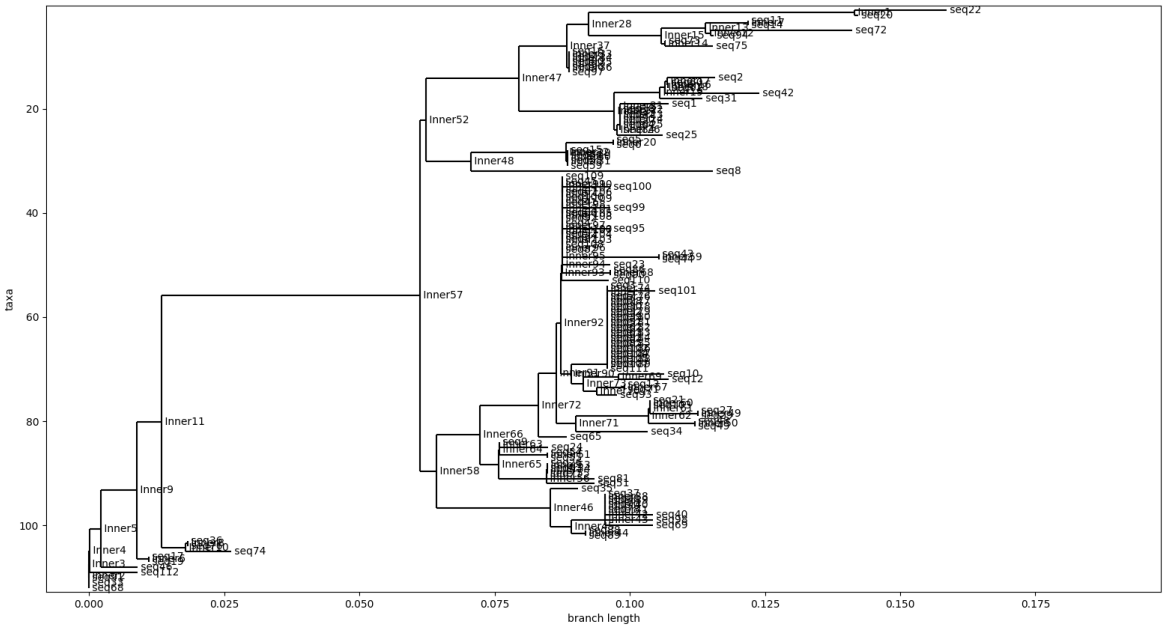


Figure 2.b : NNJ



NNJ = 101462
SPR = 203322

3. genotypes_chrM_CHB_phase3.2_consensus.b36_fwd.fasta

NNJ = 81488
SPR = 163084

4. genotypes_chrM_CHD_phase3.2_consensus.b36_fwd.fasta

   NNJ= 80432
   SPR = 160506

5. genotypes_chrM_GIH_phase3.2_consensus.b36_fwd.fasta

   NNJ = 82842
   SPR = 165556

6. genotypes_chrM_JPT_phase3.2_consensus.b36_fwd.fasta

   NNJ = 82842
   SPR = 165604

7. genotypes_chrM_LWK_phase3.2_consensus.b36_fwd.fasta

   NNJ = 90696
   SPR = 181152

8. genotypes_chrM_MEX_phase3.2_consensus.b36_fwd.fasta

   NNJ = 72362
   SPR = 144294

9. genotypes_chrM_MKK_phase3.2_consensus.b36_fwd.fasta

   NNJ = 110654
   SPR = 222172

10. genotypes_chrM_TSI_phase3.2_consensus.b36_fwd.fasta

    NNJ = 80290
    SPR = 160730

As seen in the results, there is a consistent 100% increase between the NNJ algorithm and SPR algorithm

Conclusion:

Throughout my experimentation of the SNPs of mitochondrial DNA, I have found that my SPR algorithm has consistently doubled the score of the normal NNJ tree. This means that the SPR was not better than the NNJ tree at all. This could be due to a multitude of factors. The first being incorrect processing of the DNA sequences, which means the transformation from

SNP files to fasta files could have been done incorrectly. The second issue that comes to mind is the calculation of the scores themselves. The scores seem to be extremely high for small sequences. This could be due to the number of sequences available, but they still seem too high. Another error is the reconstruction of the tree for SPR. Though there are methods to completely delete nodes, there doesn't seem to be a proper way through bio python that I could figure out to reinsert it, so I had to resort to other methods. One last error that could have occurred was the construction of the initial matrix for the phylogeny tree. In conclusion the randomness of the SPR algorithm did not surpass the consistency of the NNJ algorithm.