# SANA Spring Report 2019

*CS 199 - SANA group*

Utsav Jain, Tingyin Ding

## Abstract:

A phylogenetic tree is a diagram that represents evolutionary relationships between organisms. The branching of the tree shows how particular species evolved from a series of common ancestors. Two species are more related if they have a more recent common ancestor, and less related if they have a less recent common ancestor.

The Dual expanding Dijkstra's algorithm can be applied to biological network alignment. The edge coverage that the dijkstra based aligner gets is approximately half of SANA's coverage, but this edge coverage can be improved by adding randomness in popping an aligned pair from the skip list data structure. This semi-random method of popping pairs gives the program a chance to select the poorer similarity pair instead of always selecting the pair with the highest similarity.

## Introduction

We added command line arguments to our program that can construct phylogenetic trees, in which the user is now able to select the type of score, type of graph, and which sana output to use.

In order to compute resnik scores faster, we found a resnik program implemented in Java which is significantly faster than the python implementation. There was some discrepancy between the Java and python results, so we compared the differences between their outputs.

We started to continue the work on the Dijkstra Based aligner, which was left off by Yisheng Zhou (Joseph). We first replicated the result with the yeast and human networks which were used by the previous researchers. We then altered the program to be able to select which species to use, as well as the delta value (level of randomness). Then, we compared the EC scores produced by SANA and dijkstra algorithms. We also ran the dijkstra algorithm with varying delta values on the GHOST, graphlet, and IID sim files. For each delta value, we ran the algorithm 10 times.

## Methods

In order to run the dijkstra based alinger on different similarly files with varying delta values, we used the SGE program to run them all at once. We generated the dijkstra algorithm run commands using python scripts and piped them into /home/sana/bin/distrib_sge. The similarity files were selected from one of the three directories listed previously, and its corresponding network files were selected from /home/sana/networks/. We ran the algorithm with each sim file with a specific delta value (0.01, 0.025, 0.04, 0.0625, 0.09, 0.1225, 0.16), and

we ran it 10 times for each delta value. After producing those outputs, we wrote python scripts to visualize the results using matplotlib.
Example run: /home/sana/bin/distrib_sge | python3 run_dijkstra.py -g1 /home/sana/networks/IIDyeast.el -g2 /home/sana/networks/IIDcow.el -s /home/sana/sims/Importance/IIDyeast-IIDcow.sim.xz -d 0.025

The results were saved in this specific format: IIDyeast-IIDcow-0.025-9.51-5.dijkstra
Where IIDyeast and IIDcow are the species selected, 0.025 is the delta value selected, 9.51 being the EC score result, and 5 meaning the 5th algorithm run for this specific delta value.

## Results

### Phylogenetic Trees:

*(The scripts are located in /extra/wayne0/preserve/resnik/trees)*
In order to construct a phylogenetic tree, a distance matrix must be generated first. To create the distance file, the following command can be used:

```
python3 distance.py -t [scores: EC(default), S3, or ICS] -g [G1(default) or CCS0] -f [files]
```

Example command:

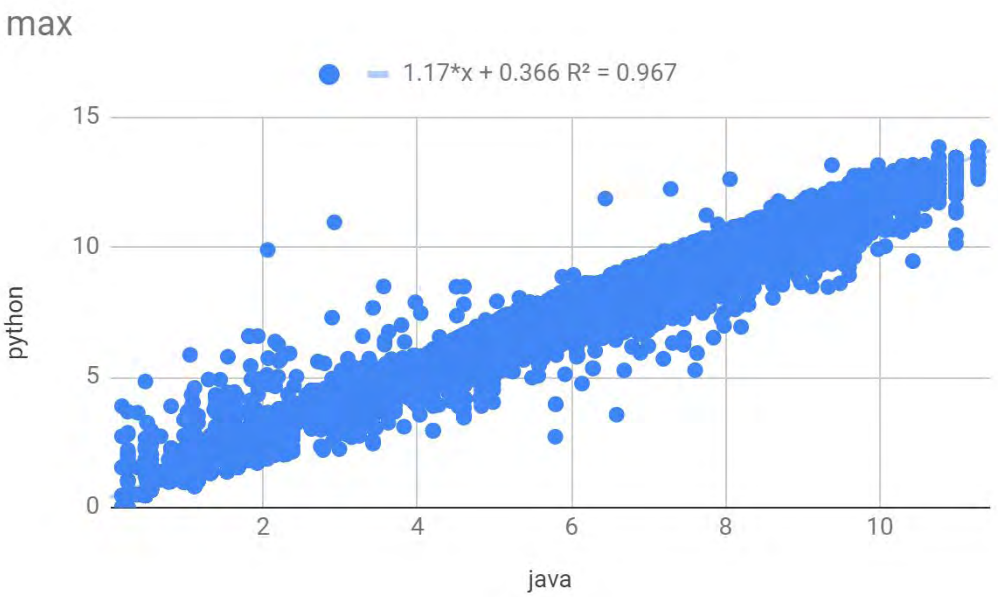
```
python3 distance.py -t EC -g G1 -f
/extra/wayne1/src/bionets/SANA.github/Jurisica/Migor/IID/GO-term-species/rat-human-s3.
t256.out  /extra/wayne1/src/bionets/SANA.github/Jurisica/Migor/IID/4h/cat-horse-s3.out
/extra/wayne1/src/bionets/SANA.github/Jurisica/Migor/IID/GO-term-species/worm-dog-s3.
t002.out
```

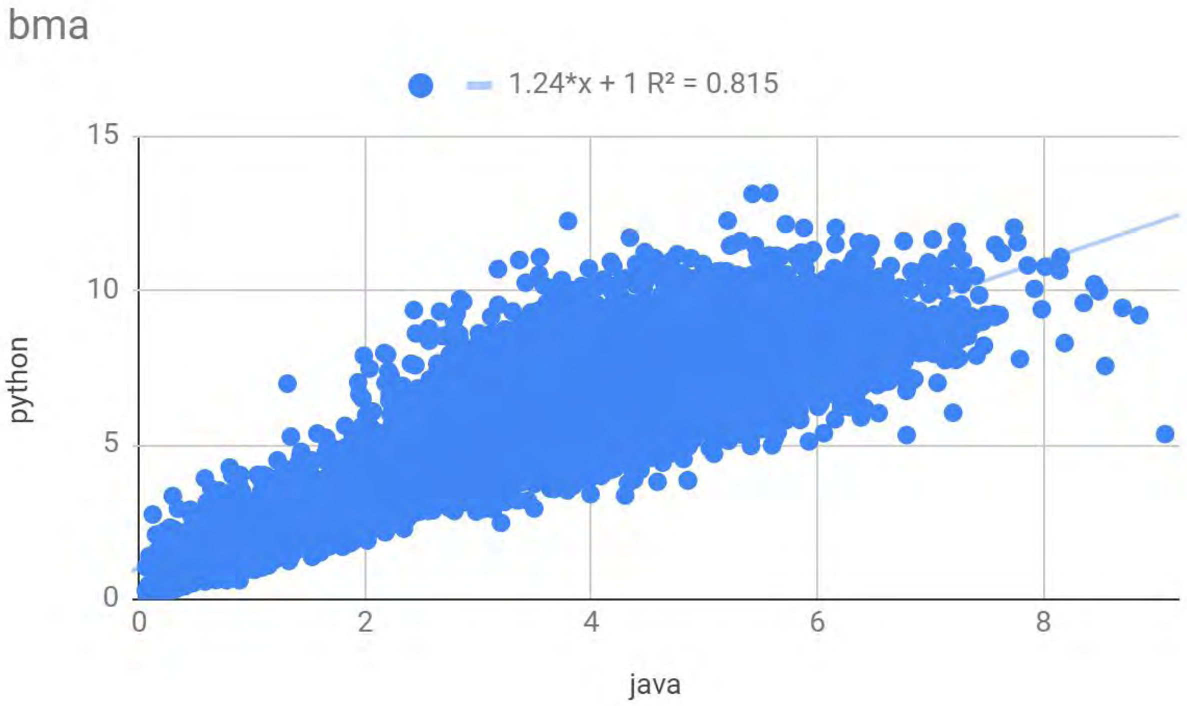
The command for generating the phylogenetic tree is:

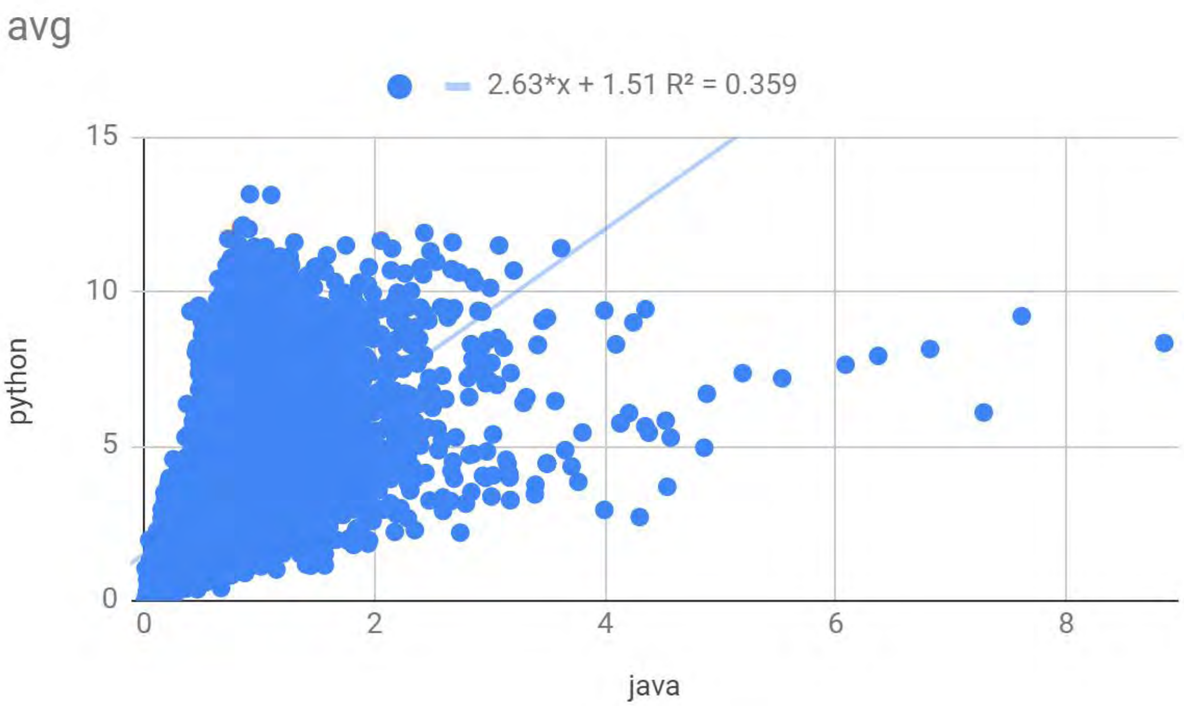
```
python3 trees.py -f [distancefile] -t [NJ or UPGMA]
```

### Java Resnik:

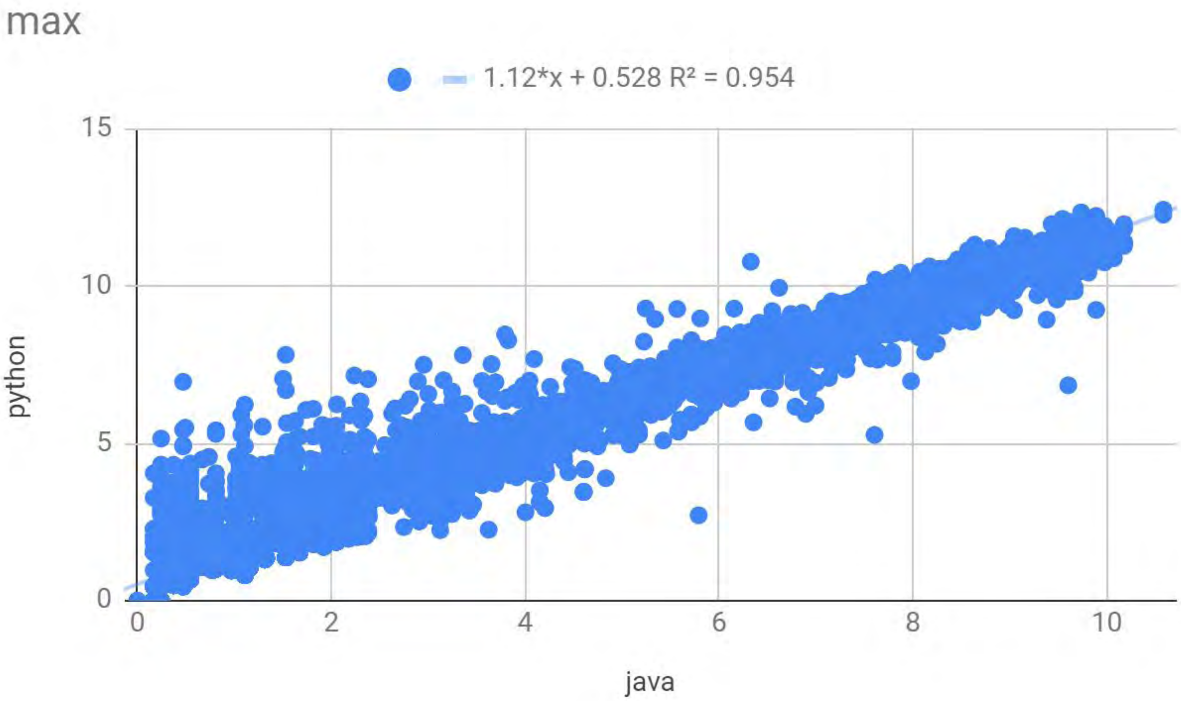
*(The script is in /extra/wayne0/preserve/resnik/java)*
The results show that max is the best mode for java resnik to produce consistent results with the python resnik program. However, there are still some deviations which are shown below in some sample comparisons.
rat-mouse-t512

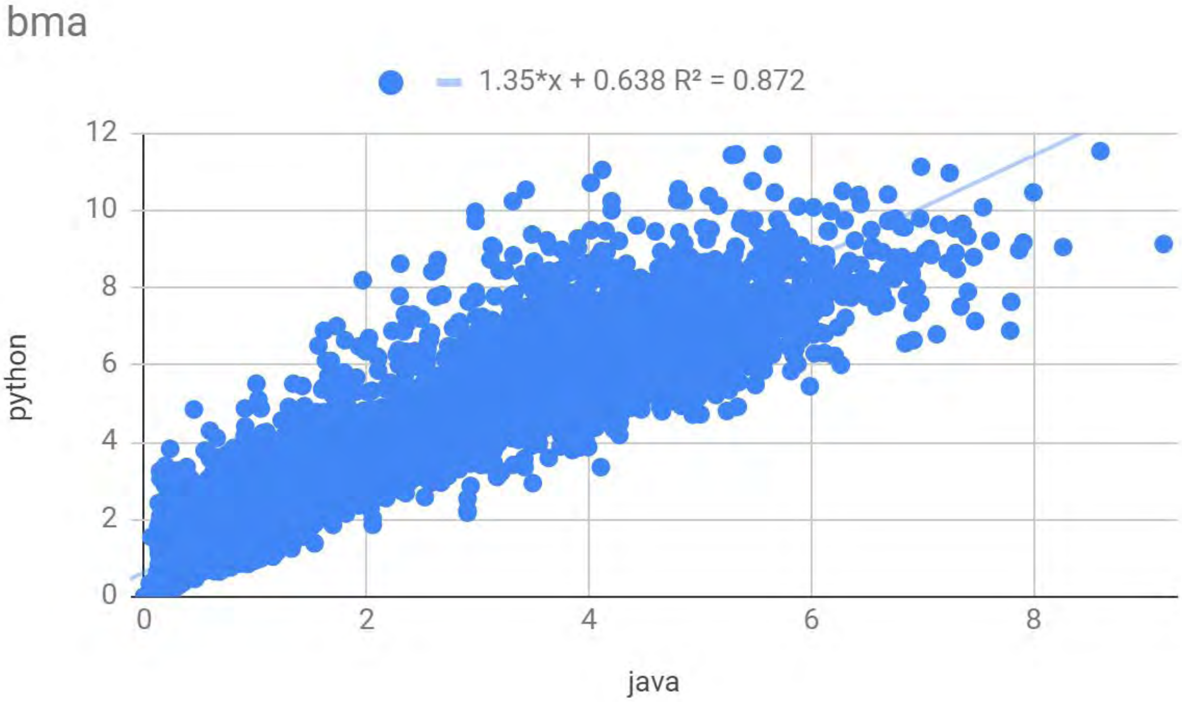

max
1.17*x + 0.366 R² = 0.967
python
java
15
10
5
0
2
4
6
8
10

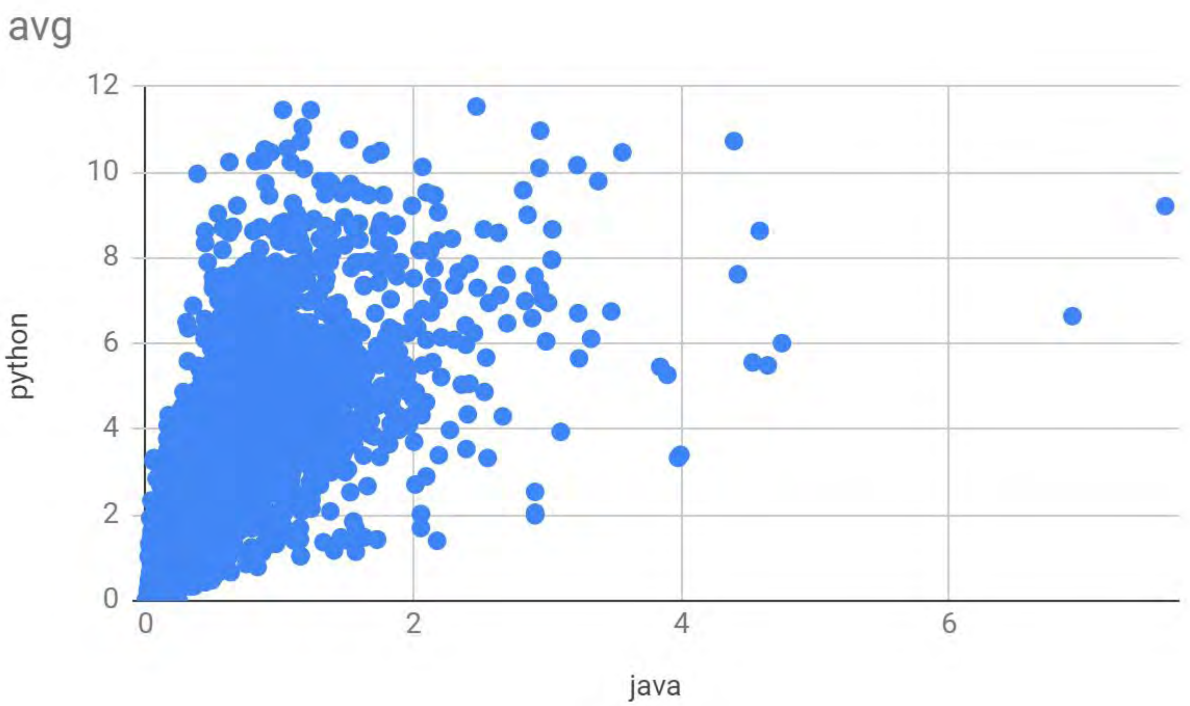

bma
1.24*x + 1 R² = 0.815
python
java
15
10
5
0
0
2
4
6
8


avg
●
2.63*x + 1.51 R² = 0.359
15
10
5
0
python
0
2
4
6
8
java

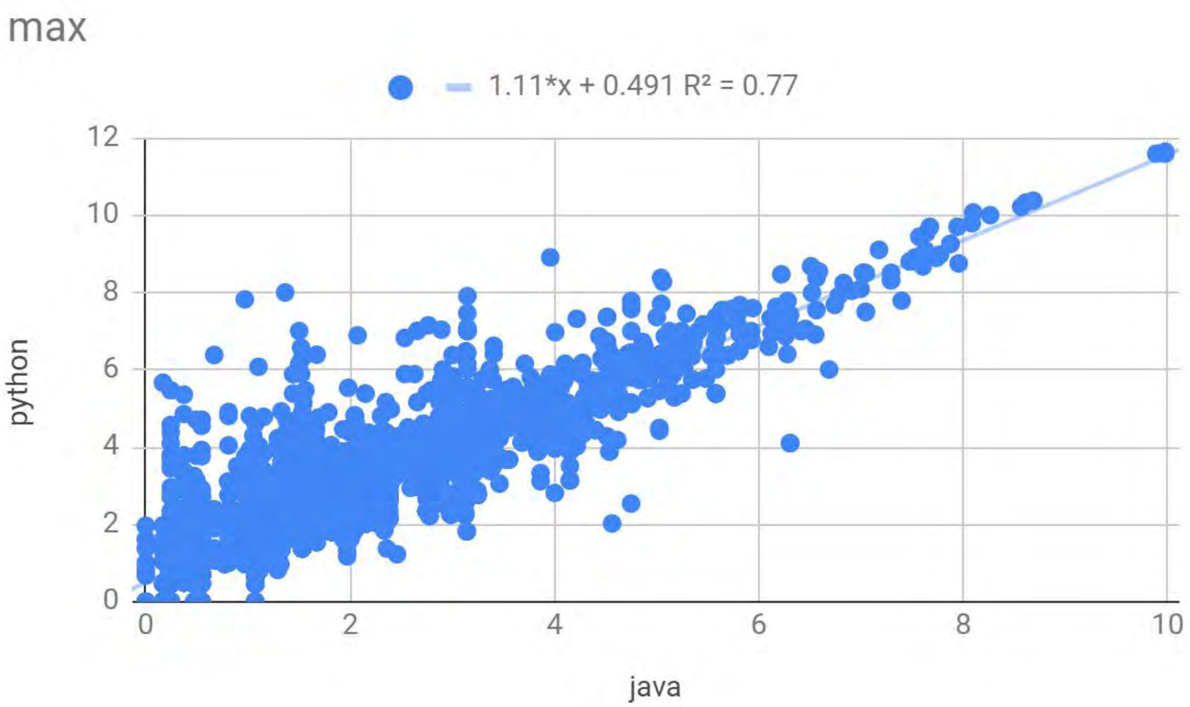
fly-cow-t512

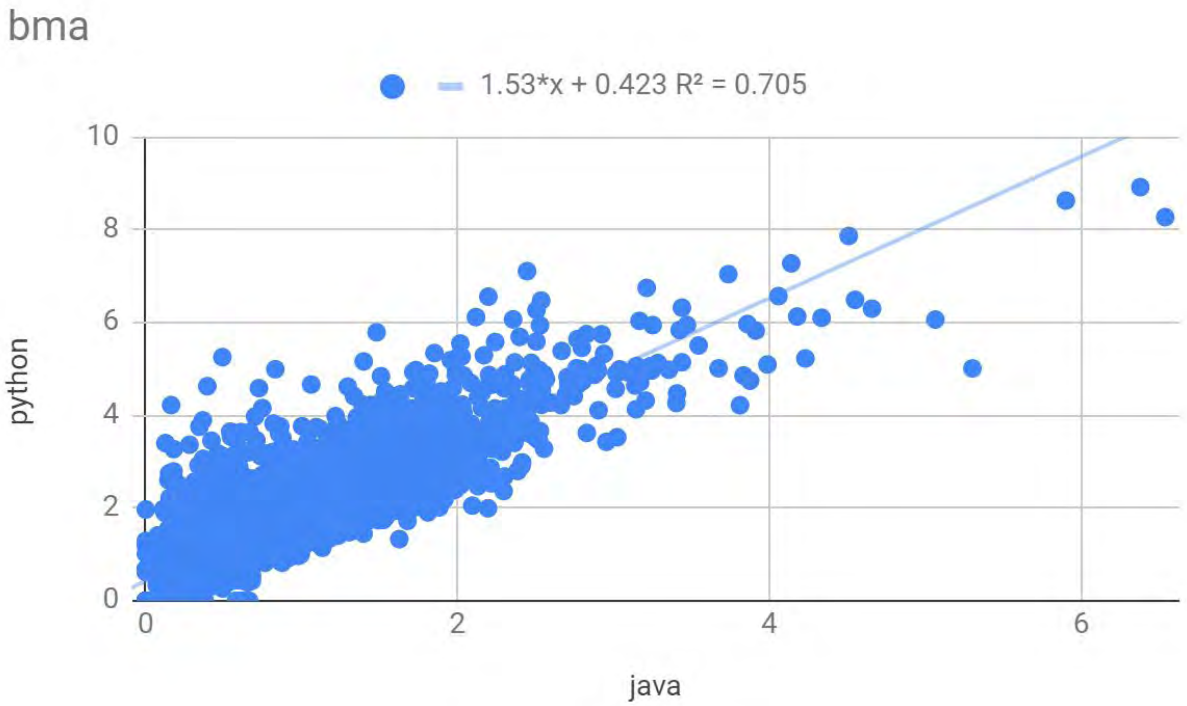

max
● 1.12*x + 0.528 R² = 0.954
15
10
5
0
python
0
2
4
6
8
10
java

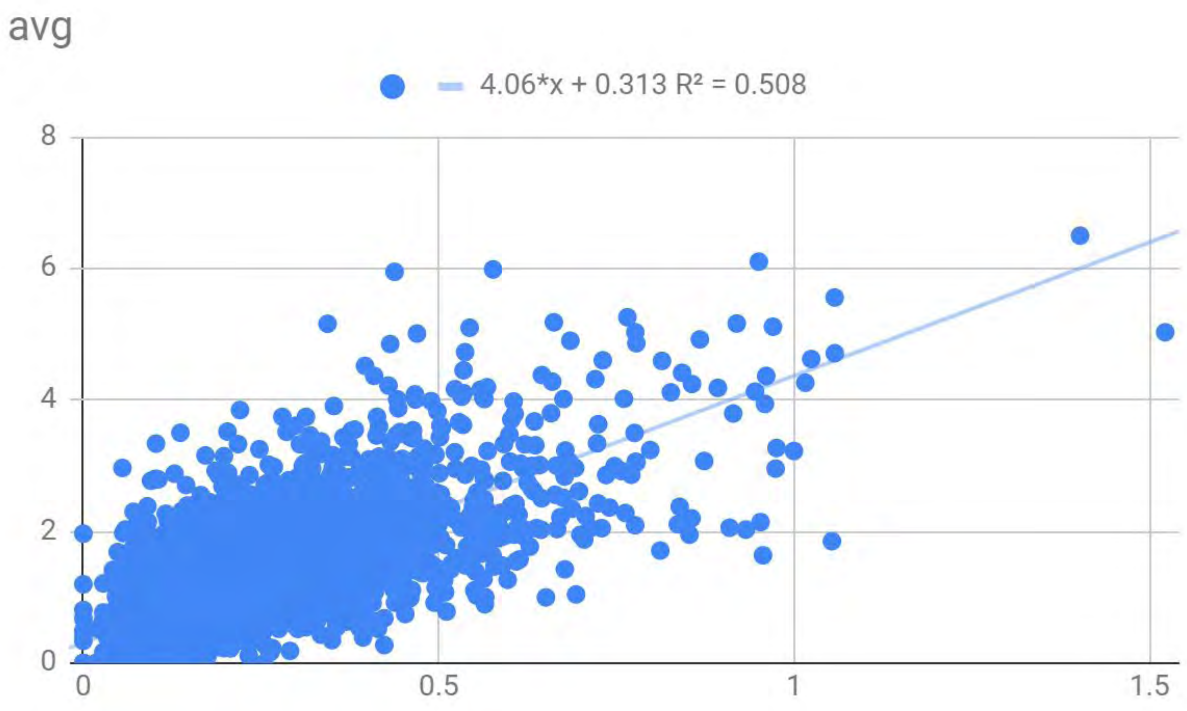

bma
● 1.35*x + 0.638 R² = 0.872
12
10
8
6
4
2
0
python
0
2
4
6
8
java



yeast-worm-t512



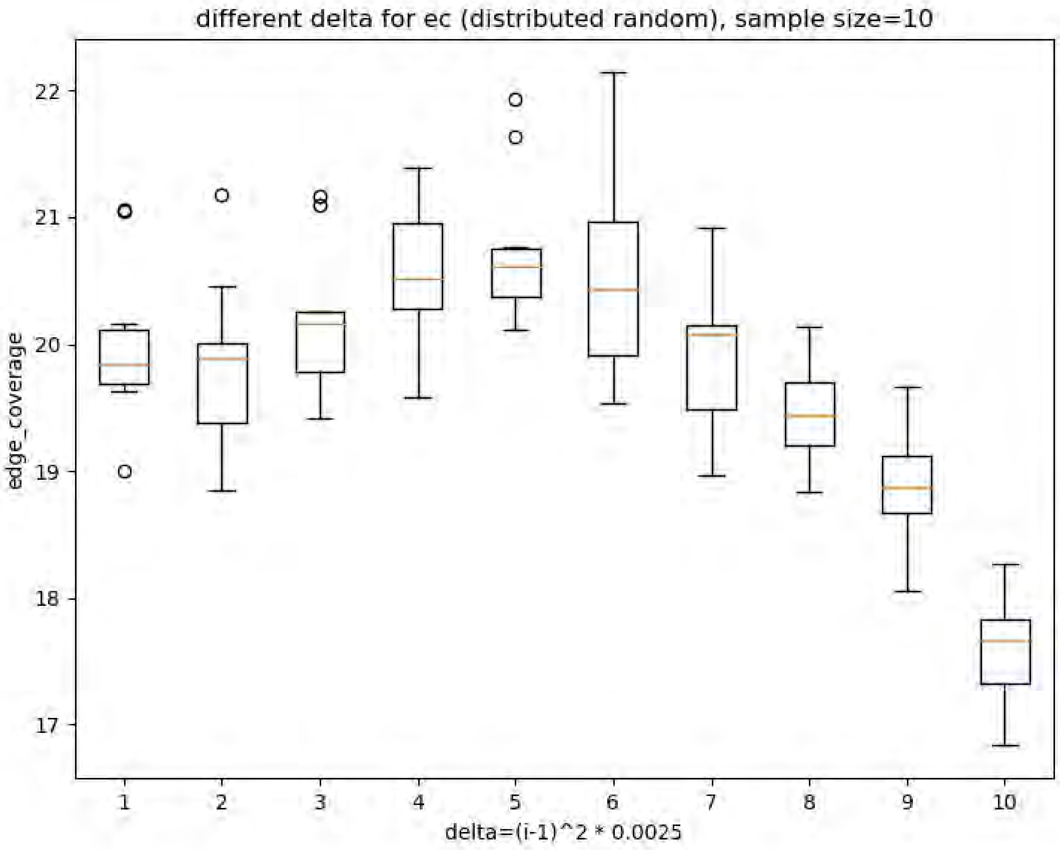

bma
1.53*x + 0.423 R² = 0.705
10
8
6
4
2
0
python
0
2
4
6
java


avg
4.06*x + 0.313 R² = 0.508
8
6
4
2
0
0
0.5
1
1.5

**Dijkstra Aligner:**

*(The scripts are in /extra/wayne0/preserve/tingyind/dijkstra/)*

1. Since the project was shelved for some time, we went through the previous team's write ups and code in order to reproduce their last results to continue where they left off. We reran the program with the sample networks, human and yeast. The result produced was similar to Joseph's.



2. Comparision with SANA's analysis

*(The raw result is in /extra/wayne0/preserve/tingyind/dijkstra/sana_analysis/analysis/result.txt)*

Besides MMusculus-SCerevisiae, SCerevisiae-AThaliana, SCerevisiae-DMelanogaster, and SPombe-CElegans, the EC scores we produced from Dijkstra were similar to SANA's analysis.
Results:
[https://docs.google.com/spreadsheets/d/1NHh_gJn_4XOWbWueG0icQjOSmgvIaM606PP3JrVFUps/edit#gid=405653798](https://docs.google.com/spreadsheets/d/1NHh_gJn_4XOWbWueG0icQjOSmgvIaM606PP3JrVFUps/edit#gid=405653798)

3. Run with different sims file and varying delta values

a) /home/sana/sims/graphlet
*(The raw outputs are in /extra/wayne0/preserve/tingyind/dijkstra/graphlet/results)*
For most alignments, the added randomness improves ec score. Only SCerevisiae-HSapiens goes down as the randomness increases. For some other alignements, the coverage goes down at first (delta = 0.0025 and delta = 0.01) but goes up ultimately when delta becomes larger.

b) /home/sana/sims/GHOST
*(The raw outputs are in /extra/wayne0/preserve/tingyind/dijkstra/GHOST/results)*
For some certain alignments (SCerevisiae-HSapiens, SCerevisiae-DMelanogaster, MMusculus-HSapiens, DMelanogaster-HSapiens, CElegans-HSapiens, AThaliana-HSapiens, and AThaliana-DMelanogaster), the results are similar to graphlet's, in which the edge coverage is improves as the delta increases. However, for RNorvegicus-CElegans, RNorvegicus-DMelanogaster, RNorvegicus-MMusculus, and RNorvegicus-SPombe, the coverage decreases. For other alignments, the edge coverage is at a peak is with no randomness or very tiny delta values, drops down with low delta values, and begins to increase again with higher delta values but never quite reaches its initial peak.

c) /home/sana/sims/Importance
*(The raw outputs are in /extra/wayne0/preserve/utsavj/ImportanceResults/)*
Although the data is incomplete right now, it is clear to see that for most species the edge coverage begins to rise as delta goes up, and reaches its peak around the delta values of 0.0625 and 0.09. After 0.09, the increasing delta value causes the edge coverage to go down. The size of these similarity files causes the algorithm to take significantly longer, and more runs for different similarity files are needed to verify this trend of the edge coverage peaking around these specific delta values. So far this trend is shown in the species pairs starting with IIDyeast, IIDworm, IIDsheep, and IIDchicken. Meaning that the pairs IIDyeast-IIDworm, IIDyeast-IIDcow, IIDyeast-chicken, etc. reflect this trend.

## Discussion

In order to prove that the Dijkstra Aligner can improve its output score by adding more randomness, we ran the algorithm with similarity files in graphlet, GHOST, and Importance with varying delta values, and with 10 runs for each delta value.

From these results, its seems promising that adding randomness to the Dijkstra Aligner can actually improve its edge coverage, especially with the graphlet and Importance similarity files. To further prove that this technique is valid, we can test different methods of injecting randomness to other aligner algorithms, and see if the edge coverage improves for them by testing them with varying delta values. This idea can be expanded to an even more broader perspective of proving that adding randomness to non-deterministic algorithms can improve them.