Sets of Protein Sequences from interesting Genomes

**Blastp 🡪 Forward Best Hit 🡪 Save Data**

**Read Data 🡪 Expansion \_ Matrix 🡪 Inflation Matrix**

**Parallel\_Computation**

**Read Data🡪 Backward Best Hit 🡪 Save Data**

**Clustering the Ortholog**

ReMark : Ortholog Cluster

**List of Functions Used in Ortho\_Detection V-1\_Old**

1. **GetMatrixNumber()**

Receive the info about matrix Number and Return the Matrix name Like "\n1. BLOSUM45\n2. BLOSUM62\n3. BLOSUM82\n4. Quit". BLOcks Substitution Matrix (BLOSUM) is a Substitution matrix used for sequence alignment of Proteins. BLOSUm45 is for more distantly related Proteins alignment Database 2. BLOSUM62 : MidRange Seq with more than 62%s Similarity. 3. BLOSUM82 :- More Related Proteins

1. **GetQuerySequence(genome):**

Receive genome Fastaq File and return gene\_sequence\_list

1. **WriteQuery(query, Parallel\_num)**
2. **RunBlast(subject, Parallel\_num)**
3. **Get\_Same\_Species\_Forward\_Best\_Hit(blastp\_score)**
4. **GetForwardBesthit(blastp\_score)**
5. **DivisionParallelQuery(queryV,query\_division\_value, cpu\_count, queryV\_len)**
6. **RunParallelQuery(species\_of\_query , species\_of\_subject , queryV, parallel\_num)**
7. **Oneway\_Threshold\_Best\_hit(mode)**
8. **Backward\_Best\_Hit(args)**
9. **Search\_Equal\_BBH\_Data(target\_A)**
10. **Search\_Unequal\_BBH\_Data(target\_B)**
11. **Matching\_BBH(target)**
12. **Generating\_Matrix\_Clustering\_Ortholog(element\_set, bar)**
13. **Parallel\_MCL(score\_matrix)**
14. **MCL(score\_matrix)**
15. **Clustering(row\_data, col\_data, score\_matrix)**
16. **Parallel\_Matrix\_Multiplication\_Using\_Numpy(data)**
17. **Parallel\_Matrix\_Power\_Using\_Numpy(matrix\_element)**
18. **Parallel\_Matrix\_Divide\_Using\_Numpy(data)**
19. **Read\_Species\_List(pr=0)** This Function is Only to display the name of Species inside the species Folder and return selected\_species\_dic (dict) , backward\_selected\_species\_dic, and total files inside species folder

(selected\_species\_dic, backward\_selected\_species\_dic, number\_i)

1. **Del\_File(path, file)**
2. **Check\_File(File)**
3. **Read\_Equal\_BBH(path)**
4. **Read\_Unequal\_BBH(path)**
5. **Read\_species\_List()**

**Blastp\_score is output file produced by runblast()**

**Blastp\_score = RunBlast(selected\_species\_dic[Species\_of\_object, parallel\_num]**

**~~Parallel\_num is number of CPU~~**

**#Steps of Program to run**

1. **Read\_Species\_List(pr =1) Run first and return 3 values**
2. **GetMatrixNumber() 🡪 blastp\_matrix**
3. **Check\_file(Cluster\_out) # Cluster\_out = command\_options.Cluster\_out**
4. **Del\_File(Score\_file, “\*”) # Score\_File = command\_options.Score\_file**
5. **Starting Log file Write**
6. **Backward\_best\_hit\_work\_list = Oneway\_Threshold\_Best\_Hit(mode) # mode = command\_options.mode**
7. **Read\_Equal\_BBH(Score\_file+selected\_species\_dic[i]+”\_”+selected\_specied\_dic[k])**
8. **Read\_unequal\_BBH(Score\_file+selected\_species\_dic[i]+”\_”+selected\_species\_dic[k])**
9. **Matchin\_BBH(unequal\_RBH\_element)**
10. **Generating\_Matrix\_Clustering\_Ortholog(data, bar)**

**## variable Name with Running Process**

1. **Mode:. Is user input to select mode among 3 that is Blastp , Blastp Using precalculated data and Clustering**
2. **Second Variable is to select Name of Genome to analyze (Suppose Only one Selected)**

**Genome\_name selected is passed to selected\_number and converted to sorted set and also program exit if input greater than number of Genome available**

1. **Blastp\_matrix = GetMatrix() Function run and Output is “BLOSUM45” , “BLOSUM62” or “BLOSUM” default is BLOSUM62**

**BLOSUM80 is used for closely related alignment databases, BLOSUM45 is used for more distantly related alignments BLOSUM62 is the Matrix built using sequences with more than 62 % Similarity (Sequences with >= 62 %identically were clustered. Note BLOSUM62 is the default matrix for protein BLAST. Experimentation has shown that the BLOSUM-62 matrix is among the best for detecting most weak protein similarities.**

**user\_selected\_number = [backward\_selected\_species\_dic[ele] for ele in genomes]**

**If mode 1 Passed**

1. **Backward\_best\_hit\_work\_list = oneway\_Threshold\_Best\_Hit(mode)**
2. **Pool = multiprocessing.Pool(cpu\_count)**

**Description of All Variables**

1. **Mode** = User input to select Mode (1 is for Blastp. 2. Blastp using precalculated data 3. Clustering

**1**

**Selected\_species\_dic** = Dictionary value of all species inside species Folder **1**

1. **Backward\_selected\_species\_dic** = Same as 2 But the Dictionary is Opposite

**Number\_i** = length of File inside species Folder **4**

1. **Selected\_number** = Number to select Gene like 1.AAE Gene Position in the Folder. This value is Used to take Out Species
2. **cpu\_count** = The Value of CPU For Parallel Counting (For Our Training We Only Use 1) \
3. **score\_file** = command\_options.score\_file (User input File)
4. **queryV =** GetQuerySequence(selected\_species\_dic[i]) . List format of sequence with a specific Gene
5. **queryV\_len = len(queryV)**
6. **backward\_best\_hit\_work\_list =** Oneway\_Threshold\_Best\_Hit(mode) (Return Value by the Function)

**Important Note**

**PAM 🡪** Point Accepted Mutation

BLOSUM

**Different Between PAM & BLOSUM**

1. PAM matrices are based on an explicit evolutionary model, whereas the BLOSUM matrices are based on an implicit model of evolution.
2. The PAM matrices are based on mutations observed throughout a global alignment, this includes both highly conserved and highly mutable regions. The BLOSUM matrices are based only on highly conserved regions in series of alignments forbidden to contain gaps.
3. The method used to count the replacements is different: unlike the PAM matrix, the BLOSUM procedure uses groups of sequences within which not all mutations are counted the same.
4. Higher numbers in the PAM matrix naming scheme denote larger evolutionary distance, while larger numbers in the BLOSUM matrix naming scheme denote higher sequence similarity and therefore smaller evolutionary distance. Example: PAM150 is used for more distant sequences than PAM100; BLOSUM62 is used for closer sequences than BLOSUM50.

**Reciprocal Best Hit algorithm (RBH)**

Reciprocal Best Hits(RBH) are a common proxy for orthology in comparative genomics.Essentially, a RBH is found when the proteins encoded by two genes, each in a different genome, find each other as the best scoring match in the other genome. NCBI’s BLAST is the software most usually used for the sequence comparisons necessary to finding RBHS.

It is used for quickly finding orthologs, i.e., genes that have diverged after a speciation event and are more likely to perform a similar function in different species, as opposed to paralogs, which are the result of a duplication event and are more likely to perform a different function. The best way to do this is by considering phylogenetic trees. If for a particular gene family (a group of genes with a similar sequence)

**Threshold\_score:🡪** Threshold\_Score is the allowable range to be considered as ortholog.

If you set the Threshold score to 0 , the concept is the same as the algorithm using reciprocal best hit. The only difference is that the blastp has been performed only once.( Threshold score를 0으로 설정하면, reciprocal best hit를 이용한 알고리즘과 개념이 같게 됩니다. 다만 blastp를 한 번만 수행했다는 것이 다릅니다). Gene sequences related to reciprocal best hit are called orthologs, when the score of blastp is the best hit in both directions. In the one-way threshold best hit algorithm, the blastp is executed once to find the forward best hit (AAE Gene1 🡪 ECO Gene1) , and the data with the reverse query type gene(ECO gene1) is found in all calculated blastp datasets. After find the backward best hit. If the difference between the backward best hit score and the forward best hit score is less than or equal to the threshold score, it is considered as ‘ortholog’, and this is called the “one-way threshold best hit”

**MCL – a Cluster algorithm for graph**

**T**he MCL algorithm is short for the Markov Cluster Algorithm, a fast and scalable unsupervised cluster algorithm for graphs (also known as networks) based on simulation of (stochastic) flow in graphs. The algorithm was invented/ discovered by Stijn Van Dongen at the Centre for Mathematics and Computer Science (also known as CWI) in the Netherlands.

**Mathematics behind the MCL Process 🡪** Expansion, which is just normal matrix multiplication, belongs to the language of linear algebra.

**Limitations of MCL**

Ofcourse , the MCL algorithm is not a panacea, and has limitations as well. Problem instances in which the diameters of the clusters are not too large allow a regime of pruning while maintaining the quality of the clusterings retrieved.

<https://micans.org/mcl/index.html?sec_thesisetc> Detail about MCL limitations can be read on the site.