Creating an efficient compression based DNA sequence clustering algorithm for the analysis and comparison of metagenomes

Abstract

DNA clustering algorithms allow the comparison and analysis of metagenomes (environmental genomes) that contain unknown and uncultivated microorganisms otherwise difficult to analyze using traditional genomic methods. Although successful, these algorithms lack the efficiency and accuracy needed to push Metagenomics as a common and accessible lab procedure. The compression based clustering approach seeks to address these gaps. By applying a lossless compression algorithm to the DNA sequences the size of these sequences can be reduced allowing increased efficiency. By creating a similarity method attuned to the compressed sequences, these compressed sequences can be clustered using any current clustering method. We demonstrate the accuracy and diversity of compression based clustering by testing six sample algorithms created by this approach, extensively over a wide range of data sets, concluding that compression based clustering has the potential to bridge the gaps in current clustering algorithms.

Introduction

Metagenomics, also known as "environmental genomics", is the sequencing and analysis of genetic material from environmental samples. The comparison of different samples provides an alternative to traditional single- genome studies for exploring the microbial world. As it allows the ability to distinguish hidden diversity in microscopic samples, Metagenomics provides powerful insight into our world and various diseases. Because the DNA is sequenced from environmental samples, most of the microorganisms found in these samples are unknown and possibly "unculturable". Even well-studied traditional genomic sequencing methods are of no use, as the clones need to be cultivated; if the clones are not recognized, traditional sequencing methods provide no insight [1]. By allowing the sequencing of uncultured genomes directly from environmental samples, Metagenomics offers new ways to study this unexplored diversity [2]. Through the study and comparison of different samples we can determine the content, abundance and functionality of the different microbes within the environmental samples, which is critical for understanding the pathogenic or symbiotic role played by these microbes. Current sequencing technologies produce short DNA fragments, called reads, from random positions in the genomes [3]. Figure 1 shows the two common approaches for analyzing reads. The shotgun approach involves blasting these reads to a database of cultivated microorganisms to find matches and extrapolate information based on these matches. However since most of the microorganisms found in environmental samples are unknown and uncultivated, the database used in the shotgun

approach is of no use. The 16S approach involves grouping these reads into clusters or OTUs (Operational Taxonomic Unit). Each OTU contains similar reads, part of the same species or subspecies. After grouping these reads, the entire OTU can be compared to known species/subspecies in an OTU database for identification. The 16S approach provides an advantage in that even if single microorganisms are unknown or uncultivated, the entire OTU will almost always be identifiable [4].

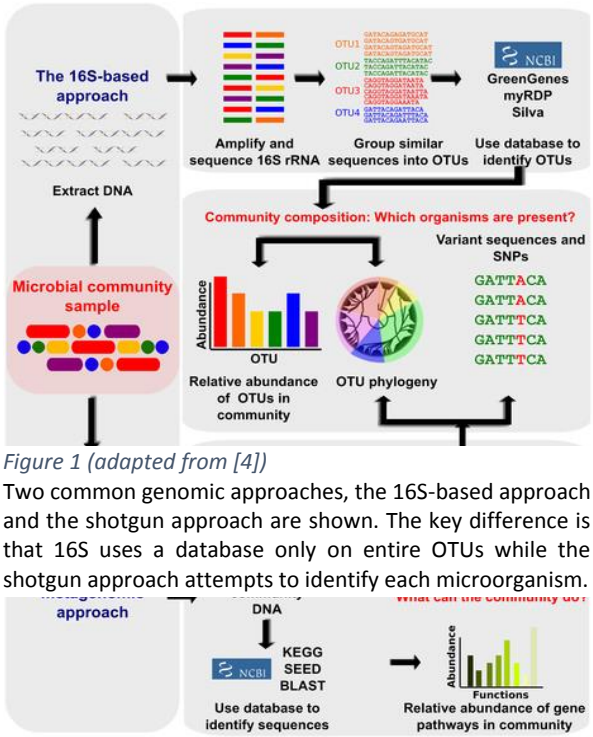

Figure 1 (adapted from [4])
Two common genomic approaches, the 16S-based approach and the shotgun approach are shown. The key difference is that 16S uses a database only on entire OTUs while the shotgun approach attempts to identify each microorganism.

Clustering methods have been developed for the 16S approach that process a large set of sequence reads of unknown origin. Clustering approaches like CD-HIT [4], UCLUST [5] and CROP [6] put similar sequences in the same groups (maybe species-specific) and then cluster representatives can be used for the rapid analysis of species diversity within different metagenome samples. These approaches use a greedy algorithm and use pairwise sequence alignments in order to compute sequence similarity. Different tools such as DOTUR [7], Mothur [8] and ESPRIT [9] improve the process of comparing metagenomes samples by using hierarchical clustering to partition the input data into clusters. In a previous work, my mentor and a graduate student created MC-MinH and MC-LSH, two greedy algorithms that used min-wise hashing and local sequence hashing in order to cluster.

In this paper, I will present a compression based approach of clustering large sets of DNA sequences into similar OTUs. My approach involves a three step process: compress the DNA sequences, identify a relevant method of computing similarity between two compressed sequences,

and apply any clustering method that is relevant to the compression algorithm to the compressed sequences using the similarity method. In order to fit the criteria for my approach, the similarity measure and clustering method need to make sense in context of the compression algorithm used. In order to validate the efficiency and accuracy of my approach, I created six algorithms using my approach and tested their performance. The algorithms created are:

- MC-Lz4DM (Metagenome Clustering using Lz4 based Dissimilarity Measurements)
- MC-SeqDM (Metagenome Clustering using Sequitur based Dissimilarity Measurements)
- MC-Lz4CD (Metagenome Clustering using Lz4 based Compression Distances)
- MC-RLEDM (Metagenome Clustering using RLE based Dissimilarity Measurements)
- MC-SeqCD (Metagenome Clustering using Sequitur based Compression Distances)
- MC-RLECD (Metagenome Clustering using RLE based Compression Distances)

The key characteristic of these algorithms is their use of compression algorithms as part of the pre-computing step. The DNA reads are first compressed into smaller more manageable sequences. Once compressed, the reads can be clustered with any clustering method. In order to cluster, the algorithm needs to determine the similarity between any two sequences.

For the purpose of this experiment, I chose two different measures of similarity: compression based dissimilarity measurement, which involves compressing the concatenation of the two reads and comparing the concatenation to the individually compressed reads, and compression distance, which uses pairwise sequence alignments similar to CD-HIT and UCLUST on the compressed reads to determine a similarity index.

I evaluated the algorithms on two different data sets as a preliminary stage in order to assess whether they meet the criteria of my approach. After weeding out the algorithms that did not contain similarity and clustering methods that were relevant to the compression algorithm, I

tested the remaining algorithms on a 16S environmental data set. If these algorithms performed as well or better than the leading algorithms in terms of run time, accuracy, and diversity then my approach is valid and shows the possibility of increased efficiency. The results are particularly strong for MC-Lz4DM, showing it to be an efficient and viable method when compared to the above algorithms. There is evidence to support future generalization of this compression based strategy into other clustering algorithms such as the ones mentioned above for increased performance.

The efficiency of comparing metagenomes is a key part of understanding the role that microorganisms play in everyday situations. Metagenomics is a very slow and often inaccurate process, making it not viable as a rigorous method for complicated observations. Disease ridden organs contain many unknown microorganisms. By comparing the environment inside a healthy and diseased organ, we can analyze the importance of microorganisms in these diseases. In order to better understand the role microorganisms play in our life, metagenomics needs to become a more viable method of observation for complex samples.

Methods

In this section we will identify my approach to clustering using a compression based algorithm. The compression function allows us to reduce the complexity of the reads allowing faster clustering. Initially I chose three compression methods to test. We will review these methods and then describe the similarity measurements I developed.

Sequitur:

Given a DNA string s of length n, Sequitur creates a set of strings $R = \{R_0, R_1, R_2, \ldots, R_m\}$ of which R_0 is the sequitur representation of s composed of terminals (A,G,C,T) and non-

terminal symbols (R_1,R_2, … ,R_m) that refer to rules R_1-R_m. String s is read character by character and used to build up R_0 in the following fashion:

Let n = 1
As each new input character is observed, append it to rule R_0 and link it to the previous symbol in R_0.
Each time a link is made between two symbols
    if the new digram (two consecutive symbols) is repeated elsewhere and the repetitions don't overlap
        if the other occurrence is a complete rule
            replace the new digram with the non-terminal symbol that heads the rule
        otherwise,
            form a new rule R_n and replace both digrams with the new non-terminal symbol R_n and **let** n = n+1
    otherwise,
        insert the digram into the index
Each time a digram is replaced by a non-terminal symbol
    if either symbol is a non-terminal symbol that only occurs once elsewhere
        remove the rule, substituting its contents in place of the other non-terminal symbol
After every character in s is read, return R_0

Note for a dataset S = {s_1 , s_2 , … s_k}, the set {R_1, R_2 , … , R_m} will be the same for all s_i because Sequitur creates a common set of rules that is applied to the whole data set. The only difference is that each s_i will have a unique R_0.

Lz4:

Given a DNA string of length n, the Lz4 algorithm creates a byte array B of length < n to represent s. The first byte in this array is a token that is divided into 4 bit fields. The first field represents the number of literal bytes to be copied to the output. The second represents the number of bytes to copy from the already decoded output buffer. In case more bytes are needed to represent these represent the literal lengths, a value of 15 is used in either of these bit fields. This token byte is then followed by a number of optional bytes to identify the literal length if indicated by a value of 15 in the token. After this comes a string of literals which holds the compressed string. There is then two bytes for an offset which tells the algorithm where in the literals it should start copying. After the offset, the remaining bytes represent matches within the literals which tells the algorithm

about duplicates and matches allowing a higher degree of compression [10]. The format for compression is shown in Figure 2.

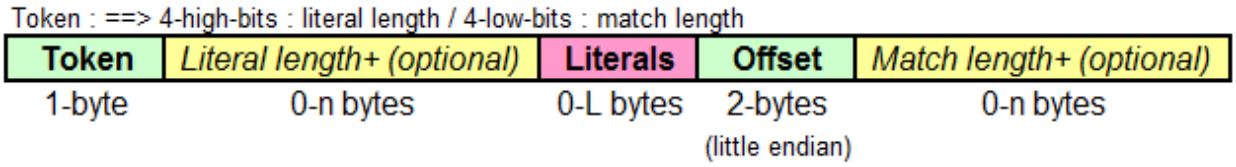

Figure 2 (adapted from [10])
A sequence compressed in the Lz4 format is split into a token, literals, offset and the match. The token indicates various lengths that need to be copied from the literals to the buffer. The literal bytes are the encoded bytes. The offset determines the start and end bytes for the encoded bytes and the match length is used to indicate which bytes match the literals in order to reconstruct the original string.

RLE:

Given a string s of length n, RLE compresses the string into string r with length < n. The final length of r depends on the number of consecutive repeating characters in s. It compresses multiple repeated characters into a character and a count. For example the sequence aaaabbccc would be compressed to a4b2c3. The algorithm is shown here:

```
Let char = null
Let count = 0
Let t = “”
for each character in the s
   if current character is not equal to char
      if count is greater than 1
          append count to t
      set char to current character
      set count to 1
   else
      increment count by 1
if count is greater than 1
   append count to t   // record last run
```

In order to cluster, we take these compressed sequences and compare them using similarity methods.

Similarity Measures:

For each algorithm we tested two different similarity methods, Compression Distance based similarity (CD) and Compression based Dissimilarity Measures (CDM).

Compression Distance:

Sequitur:

Given two strings a and b compressed by Sequitur that share a common rule set of R = $\{R_1,R_2,\ldots,R_m\}$, Let A and B be the set of non-terminal rules represented by the non-terminal symbols in a and b respectively. The Compression Distance between a and b is:

$$D = \frac{size\ of\ A \cup B}{size\ of\ A + size\ of\ B}$$

Lz4:

Given two byte arrays B_1 and B_2 compressed by Lz4 with lengths n_1 and n_2 respectively, the compression distance between B_1 and B_2 is defined as the edit distance edit(n_1, n_2) between the two arrays where

$$edit(x) = \begin{cases} \max(i,j), & \min(i,j) = 0 \\ min\begin{pmatrix} edit(i-1,j)+1 \\ edit(i,j-1)+1 \\ edit(i-1,j-1)+match(i,j) \end{pmatrix}, & \min(i,j) \neq 0 \end{cases}$$

$$match(i,j) = \begin{cases} 0, & B_1[i] = B_2[j] \\ 1, & B_1[i] \neq B_2[j] \end{cases}$$

RLE:

Given two strings s_1, s_2 compressed using RLE of length n_1 and n_2 respectively, the compression distance between s_1 and s_2 is defined as the edit distance edit(n_1,n_2). The edit function is the same as the function used for Lz4, but the match function differs as follows:

$$match(i,j) = \begin{cases} 0, & char\ at\ i\ in\ s_1 = char\ at\ j\ in\ s_2 \\ 1, & char\ at\ i\ in\ s_1 \neq char\ at\ j\ in\ s_2 \end{cases}$$

Compression based Dissimilarity Measures:

For two data quantities (Strings for Sequitur and RLE and Byte arrays for Lz4) D_1 and D_2, compressed from the DNA strings a and b using compression algorithm M, the compression based dissimilarity measure is defined as:

$$CDM = \frac{size\ of\ M(a+b)}{size\ of\ D_1 + size\ of\ D_2}\ where\ a+b\ is\ the\ concatenation\ of\ the\ two\ strings$$

Clustering:

In this section I will describe the clustering algorithm used. Assume that the clustering is being done with DNA data set S = $\{s_0,s_1,\ldots,s_n\}$ using Compression algorithm M, similarity measurement D and a threshold T which was previously experimentally derived to be optimal. The result of the clustering should produce a O = $\{O_0,O_1,\ldots,O_n\}$, the set of OTUs in S. For the purposes of my experiment, I chose to incorporate the most basic clustering algorithm adopted from CD-HIT, which randomly chooses a sequence and builds a cluster around it.

Each element s_i from S is compressed using compression algorithm M to create S' = $\{s'_0,s'_1,\ldots,s'_n\}$ where $s'_i = M(s_i)$. A random sequence s_i from S' is chosen to be a cluster representative, or the first element of cluster O_0. Then each of the remaining elements from S' is compared to the cluster representative s_i using the similarity measurement D. If the similarity measurement of the sequence and the representative is greater than the threshold T, then that sequence is considered to be part of O_0 and is removed from S' and added to O_o. From the remaining sequences in S', a random cluster representative for O_1 is chosen and the process is repeated until S' is empty. The clustering method follows a greedy iterative strategy because the cluster representative is chosen at random, and a cluster is built around it and the process is repeated until all sequences have been assigned to a cluster.

The threshold T is different for each combination of compression algorithm and similarity measure, but can be experimentally derived by evaluating clustering results on a gradient of threshold values. I derived a threshold T for four of my algorithms (MC-Lz4CD, MC-Lz4DM, MC-RLECD, and MC-SeqDM) during the evaluation.

Compression based Approach:

The approach I created in this project, combines the three aforementioned parts into one algorithm. ***Using any compression algorithm, any similarity measurement relevant to the compressed algorithms, and any clustering method, an algorithm can be created that is hypothesized to have a higher efficiency then the clustering method by itself.*** For the purposes of my experiment, I tested three compression algorithms with two similarity measurements and one clustering method. The specific combinations of these resulted in the six algorithms that I created: MC-Lz4CD, MC-SeqCD, MC-RLECD, MC-Lz4DM, MC-SeqDM, and MC-RLEDM.

Evaluation:

Preliminary evaluation stage:

Before the algorithms were used to cluster data sets, I tested the accuracy of the similarity methods of each of the algorithms. Local sequence alignment (LSA) is a rigorous method of finding the similarity between two DNA sequences that involves aligning similar partitions of the sequences to find out exactly what percent of the sequences are similar. However this is a very time consuming process, which is why it is not used in practice, even though it has a high accuracy rate[11]. For each sequence s_i in S, I identified the top five best sequence in terms of similarity based on LSA to s_i. I then repeated this process with each of my algorithms and compared the top five sequences for each s_i and recorded the number of matches.

Clustering evaluation:

The best performing algorithms from the preliminary evaluation stage were then evaluated in the secondary stage.

Secondary Evaluation stage - 16S Simulated Metagenomic samples:

The simulated data contains 345,000 short sequences, generated from 43 known 16S rRNA gene fragments using the Roche GS20 system [12]. Since this data set is simulated, an exact OTU number of 43 is expected. However since real life data samples have processing errors, data sets with 3% and 5% error were derived from the simulated data to mimic environmental conditions.

The algorithms were clustered using this data set and the number of OTUs were reported. If the number of OTUs was as close or closer to 43 in comparison with the leading algorithms, then the clustering method was considered relevant and these algorithms were passed to the final clustering evaluation.

Final evaluation 16S Environmental metagenomic samples:

This dataset contains eight seawater samples taken from a study by Sogin et. al.[13]. All of these samples have known diversity of microbes and their relative abundance in the ocean. Samples contain unequal length sequences with average sequence length of 60 characters.

The algorithms were clustered using the 6 samples from this data set and standard and diversity metrics were reported.

Results

Performance Metrics:

The performance of my algorithms were tested using standard metrics as well as diversity metrics. The standard metrics reported are number of clusters, run time and weighted accuracy.

The weighted similarity is based on the average similarity of each cluster produced by the algorithm. The weighted similarity of a cluster is measured by finding the average similarity between all pairs of sequences within that cluster. This similarity is found by taking an LSA of the pairs, because Local Sequence Alignments are considered the most accurate form of determining similarity between two sequences.

The diversity metrics were the Chao1, Shannon, and ACE diversity indexes. These indexes measure different aspects of diversity within each of the clusters classified by clustering algorithms. The Chao1 index measures the overall richness of species and subspecies along all of the clusters, where the richness is defined as the number of possible species present. The ACE index also measures richness, but it takes into account rare clusters, clusters with less than 10 members, and their implication on the number of species. The Shannon index provides a measure of entropy in the environmental sample. It quantifies the degree of surprise associated with the cluster prediction our algorithms make [14].

Implementation:

These algorithms were written in Java and tested in Ubuntu 11.04 workstation on a Windows Azure cloud. I used three virtual machine Ubuntu instances on the cloud, testing in parallel to save time and test large datasets more often.

Preliminary evaluation:

The preliminary evaluation was assessed based on the percent match between each of the six algorithms and the ground truth represented by the LSA similarity. The data set used was the 112R data set containing a total of 11132 DNA reads from the 16S Environmental set of samples. Table 1 shows that MC-SeqCD and MC-RLEDM produced an abnormally low percent match (less than 5%), indicating that the similarity measurements used were not relevant to the compression

methods. MC-SeqDM is identical to MC-SeqCD, differing only in the similarity method, but it scored 80 points higher. Hence the similarity methods used in MC-RLEDM and MC-SeqCD were not appropriate for the compression algorithms.

Preliminary evaluation using 112R data set from 16S environmental set of samples						
Algorithm	MC-SeqCD	MC-Lz4CD	MC-RLECD	MC-SeqDM	MC-Lz4DM	MC-RLEDM
Percent match with LSA similarity algorithm	2.06	91.2	84.9	86.3	86.2	.011

Table 1

Using the Local Sequence Alignment (LSA) to establish the true similarity between sequences, the similarity methods of the six algorithms were compared to LSA to determine the accuracy of these methods.

For instance, MC-RLEDM uses the RLE compression algorithm with the compression based dissimilarity measure. This dissimilarity measure is entirely based on the compressed sequence produced by concatenating the original sequences. However the RLE compression algorithm does not compress a sequence holistically, but analyzes it character by character. Therefore, concatenating two sequences and then compressing with RLE is no different from compressing first and then concatenating, unless there is a repeated character at the point of concatenation. Thus the compression based dissimilarity measure is completely irrelevant when used with the RLE compression algorithm. For the second stage of evaluation, I chose the remaining algorithms: MC-Lz4CD, MC-Lz4DM, MC-RLECD and MC-SeqDM.

Clustering evaluation:

After clustering all of the 16S Environmental and Simulated data set using various different threshold numbers, an optimal threshold for each of the algorithms were determined. Figure 3 shows the values which are from 0 to 1.0. A value of .64 indicates that for MC-Lz4DM, members of the same cluster must be at least 64% similar. MC-Lz4CD has a stricter similarity measure requiring a lenient threshold of 18%.

Algorithm	Threshold
MC-Lz4DM	.64
MC-SeqDM	.34
MC-Lz4CD	.18
MC-RLECD	.26

Figure 3

The threshold values for each of the algorithms, experimentally derived. These values determine the level of similarity needed for two sequences to be a part of the same cluster

Secondary Evaluation - 16S Simulated data set:

The 16S Simulated data set results are reported in Table 2. These artificial samples have a ground truth of 43 clusters and are organized into two samples, with an introduced 3% error and an introduced 5% error.

Algorithm	3% error # of clusters	5% error # of clusters
MC-Lz4DM	49	62
MC-Lz4CD	63	65
MC-SeqDM	220	212
MC-RLECD	240	232
MC-MinH	39	37
MC-LSH	47	41
UCLUST	91	53
CD-HIT	108	47
ESPRIT	180	86
DOTUR	210	135
Mothur	214	138

Table 2

The secondary evaluation stage was performed on artificially created data sets. These data sets had a true answer of 43 initial clusters, but were altered to reflect the 3% and 5% error in measuring produced by environmental sample sequencing machines.

For the 3% error sample, almost all of the algorithms overestimate the number of clusters, with MC-Lz4DM (49), MC-MinH (39), and MC-LSH (47) being the closest to the truth. In the 5% error sample the top contenders were MC-MinH (37), MC-LSH (41), CD-HIT (47) and UCLUST (53) while MC-Lz4DM and MC-Lz4CD were in the 60 cluster range with MC-SeqDM at 212 and MC-RLECD at 232. At this point, MC-SeqDM and MC-RLECD were eliminated from my approach as they did not fit the criteria of having a relevant clustering method. The clustering method chosen simply was not relevant in the context of the compression algorithm. Comparing the two algorithms to a ground truth algorithm shows us that they produce abnormally high values (220, 212 and 240, 232) compared to the other algorithms and therefore should not be taken to the final stage of clustering evaluation.

16S Environmental data set:

Standard Metrics: The standard metrics for the 16S Environmental data set are reported in Table 3. The number of clusters, weighted similarity and run time of my two algorithms in comparison to the rest, were the metrics tested.

MC-Lz4DM outperforms all other algorithms in terms of weighted similarity with MC-Lz4CD in a close second. In terms of speed, MC-Lz4DM finishes fourth on average with MC-Lz4CD finishing fifth. Comparable speeds and an increase in efficiency indicates that these two algorithms are viable for clustering based on the standard metrics.

Algorithm	Metric	53R	55R	112R	115R	137	138
MC-Lz4DM	# Clu	1009	868	1419	1056	849	858
	W.Sim	99.00	98.69	98.72	99.03	99.21	98.90
	Time(s)	34	24.2	23.9	24.1	34.7	27.8
MC-Lz4CD	# Clu	922	777	1302	921	754	789
	W.Sim	98.64	98.26	98.50	98.52	98.66	98.66
	Time(s)	43.7	31.9	43.6	384.4	47.4	37.6
MC-MinH	# Clu	1165	1077	1634	1156	1020	1042
	W.Sim	96.90	92.45	91.18	93.33	95.86	93.10
	Time(s)	2.5	2.1	3.3	3.0	2.7	2.5
MC-LSH	#Clu	1172	1199	1795	1205	1041	1072
	W.Sim	96.90	93.12	91.33	93.50	95.86	93.10
	Time(s)	161.0	183.0	317.0	188.0	172.0	175.0
UCLUST	#Clu	1062	992	1561	1071	900	923
	W.Sim	96.67	91.67	91.02	93.33	93.50	92.82
	Time(s)	2.0	2.0	2.0	2.0	2.0	2.0
CD-HIT	#Clu	824	716	1196	820	712	725
	W.Sim	92.56	90.80	90.61	93.33	91.82	90.16
	Time(s)	3.6	3.1	3.9	3.8	3.2	3.1
ESPRIT	#Clu	940	859	1361	970	818	832
	W.Sim	93.12	91.35	90.88	93.33	91.82	90.16
	Time(s)	283.0	266.0	537.0	348.0	280.0	296.0
DOTUR	#Clu	1241	1258	1854	1279	1096	1121
	W.Sim	96.95	94.06	91.33	93.50	95.86	93.10
	Time(s)	5129.0	3511.0	5567.0	9237.0	6563.0	5618.0
Mothur	#Clu	1238	1256	1853	1278	1094	1119
	W.Sim	96.95	94.06	91.33	93.50	95.86	93.10
	Time(s)	10130.0	5940.0	12303.0	13501.0	12861.0	12310.0

Table 3

The standard metrics was evaluated for 2 algorithms created with the compression based approach (MC-Lz4DM and MC-Lz4CD) and 7 industry algorithms on a 16S Environmental dataset containing 6 samples. The number of clusters, weighted similarity and runtime were reported.

Species Diversity:

The diversity metrics of MC-Lz4DM and MC-Lz4CD in comparison to MC-MinH are shown in Table 4. On average, both of my algorithms underestimate the richness of the clusters through the Chao1 index, but accurately measure the richness through the ACE index.

Both of my algorithms match MC-MinH in entropy as demonstrated by the Shannon Index. In general, my algorithms create clusters with comparable entropy and richness to the current leading algorithm indicating that both of my algorithms are viable clustering methods in terms of species diversity metrics.

SID	Algorithm	Chao1 Index	Shannon Index	ACE Index
53R	MC-Lz4DM	1949.1	4.3	2140.1
	MC-Lz4CD	1951.1	4.1	2424.1
	MC-MinH	2276.3	4.4	2243.7
55R	MC-Lz4DM	1766.1	4.4	2304.4
	MC-Lz4CD	1619.6	4.1	2096.7
	MC-MinH	2182.8	4.6	2214.1
112R	MC-Lz4DM	3304.4	5.1	4479.5
	MC-Lz4CD	3060.4	5.1	4296.9
	MC-MinH	3931.3	5.3	4202.7
115R	MC-Lz4DM	2087.3	4.3	2822.3
	MC-Lz4CD	1999.9	4.1	2449.6
	MC-MinH	2411.4	4.6	2455.8
137	MC-Lz4DM	1583.0	4.6	2054.3
	MC-Lz4CD	1344.3	4.4	1813.2
	MC-MinH	1992.2	4.8	1800.1
138	MC-Lz4DM	1557.2	4.2	2159.1
	MC-Lz4CD	1305.1	4.1	1942.8
	MC-MinH	1713.8	4.4	1760.3

Table 4

The two algorithms produced by the compression based approach were compared to MC-MinH, the leading industry algorithm on the same 16S Environmental dataset. The metrics reported were the Chao1 Index, a measure of richness, the Shannon Index, a measure of entropy, and the ACE Index, a more comprehensive measure of richness.

Discussion

MC-Lz4DM:

The results show that this algorithm performed the best in terms of weighted similarity compared to all of the other algorithms. This algorithm also had comparable run time with the current industry leading algorithms, placing 4th on average. In terms of diversity this algorithm

maintains the richness and entropy indexes when compared with MC-MinH, indicative of MC-Lz4DM's accuracy. This algorithm has been shown to follow the criteria of the compression based approach and is efficient and accurate. Therefore we can conclude that this algorithm is a viable scalable metagenomic sequence clustering method.

MC-Lz4CD:

Although it was not the most accurate or fastest algorithm, it was a top contender (second on average) in terms of weighted similarity and 5th in terms of run time. Similar to MC-Lz4DM, the diversity of this algorithm in richness and entropy is very similar to MC-MinH, one of the leading clustering algorithms. This algorithm also follows all the criteria of the compression based approach. MC-Lz4CD is also a viable candidate for a scalable metagenomic sequence clustering algorithm.

Implication of the results:

According to the hypothesis, since the algorithms that the compression based approach produced were concluded to be viable clustering algorithms, we have demonstrated the computational efficiency and accuracy of the compression based approach. Therefore by combining any compression algorithm, with appropriate similarity and clustering methods, we can create a viable solution.

In contrast to previous work done in the field of clustering methods, my approach provides a universal method that can be tailored to create individual algorithms. It allows the combination of currently existing clustering methods with compression algorithms to speed up and increase the accuracy of DNA clustering. Essentially, my work enhances current clustering methods by allowing the integration of different compression formats into these clustering methods.

Conclusion:

We present a universal compression based approach for creating DNA clustering methods by combining any data compression algorithm, measure of similarity between two sequences, and existing clustering method. Algorithms created using this approach were evaluated on a multitude of data samples and a comprehensive study on accuracy, diversity, and efficiency was produced. These algorithms showed a high level of efficiency, accuracy, and comparable diversity when creating clusters. We demonstrated that these results indicate the validity of the compression based approach, allowing the extension of this approach to create more efficient algorithms.

The success of this compression based approach has the potential to produce algorithms with serious improvements in terms of efficiency and accuracy.

For further testing, more algorithms created by the compression based approach need to be tested on a variety of data sets in order to assess consistency. In order to truly answer if metagenomic clustering will ever become efficient enough to be regularly used, my approach needs to be implemented with a high efficiency compression algorithm such as the ZIP file format algorithm in combination with a proven clustering method such as MC-MinH or MC-LSH.

This approach is the next step in making metagenomic clustering a more accessible method for laboratory scientists. By reducing the time and increasing the efficiency of clustering results, metagenomic clustering can be implemented in common laboratory practices allowing the easy comparison of microorganisms. Metagenomic clustering can open doors into new biological research involving microorganisms, in fields such as disease diagnostics and treatment.

References:

[1] Wooley JC, Godzik A, Friedberg I (2010) A Primer on Metagenomics. PLoS Comput Biol 6(2): e1000667. doi:10.1371/journal.pcbi.1000667

[2] Marco, D, ed. (2011). Metagenomics: Current Innovations and Future Trends. Caister Academic Press. ISBN 978-1-904455-87-5.

[3] Gene W. Tyson and Philip Hugenholtz. Metagenomics. Nature Reviews Microbiology, Sep 2008

[4] Morgan XC, Huttenhower C (2012) Chapter 12: Human Microbiome Analysis. PLoS Comput Biol 8(12): e1002808. doi:10.1371/journal.pcbi.1002808

[5] Soumitesh Chakravorty, Danica Helb, Michele Burday, Nancy Connell, and David Alland. A detailed analysis of 16s ribosomal rna gene segments for the diagnosis of pathogenic bacteria. *Journal of Microbiological Methods*, 69(2):330 339, 2007.

[6] Anne Chao. Nonparametric Estimation of the Number of Classes in a Population. *Scandinavian Journal of Statistics*, 11(4), 1984.

[7] Anne Chao and Shen M. Lee. Estimating the Number of Classes via Sample Coverage. *Journal of the American Statistical Association*, 87(417):210-217, 1992.

[8] Anveshi Charuvaka and Huzefa Rangwala. Evaluation of short read metagenomic assembly. *BMC Genomics*, 12.

[9] Sourav Chatterji, Ichitaro Yamazaki, Zhaojun Bai, and Jonathan A. Eisen. Compostbin: A dna composition based algorithm for binning environmental shotgun reads. In *In: Research in Computational Molecular Biology, Proceedings*, pages 17-28, 2008.

[10] Collet, Y (2011, May, 26). *LZ4 explained*. retrieved August 13 2013, from RealTime Data Compression Web Site: http://fastcompression.blogspot.in/2011/05/lz4-explained.html

[11] Needleman, Saul B.; and Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325

[12] Susan Huse, Julie Huber, Hilary Morrison, Mitchell Sogin, and David Welch. *Accuracy and quality of massively parallel dna pyrosequencing*. Genome Biology, 8(7):R143, 2007.

[13] Mitchell L. Sogin, Hilary G. Morrison, Julie A. Huber, David Mark Welch, Susan M. Huse, Phillip R. Neal, Jesus M. Arrieta, and Gerhard J. Herndl. Microbial diversity in the deep sea and the underexplored "rare biosphere". *Proceedings of the National Academy of Sciences*, 103(32):12115{12120, 2006.

[14] Jost, L. (2006) Entropy and diversity. Oikos, 113, 363–375. doi:10.1111/j.2006.0030-1299.14714.x