SCALABLE AND ROBUST CLUSTERING AND VISUALIZATION FOR LARGE-SCALE BIOINFORMATICS DATA

Yang Ruan

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Department of Computer Science

Indiana University

August 2014

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degress of Doctor of Philosophy.

Doctoral Committee

Geoffrey C. Fox
(Principal Advisor)

David Leake

Judy Qiu

August 18, 2014

Haixu Tang

There are no secrets to success.

It is the result of preparation, hard work, and learning from failure.

Colin L. Powell

Acknowledgements

First of all, I am sincerely grateful to my adviser Professor Geoffrey C. Fox, for his farsighted guidance, insightful advice, and unlimited support during the past years. His guidance pointed out to me the correct way of doing research as a scientist in this dissertation as well as in my other research projects. On the basis of his patience and valuable advice, I significantly increased my knowledge as I work toward the PhD degree.

It is my great pleasure to thank the members of my research committee: Professor Judy Qiu, Professor Haixu Tang and Professor David Leake. I cannot overstate their invaluable comments that helped me understand and solve the research issues. These suggestions given by them greatly expanded my vision in the area of research.

It is been a pleasure working for various projects at SALSA HPC group in Community Grid Lab. I enjoyed the weekly discussions with Professor Judy Qiu and my colleagues: Dr. Jaliya Ekanayake, Dr. Thilina Gunarathne, Xiaoming Gao, Saliya Ekanayake, Bingjing Zhang, Tak-Lon Wu, Hui Li, Fei Teng, Yuduo Zhou, Jerome Mitchell, Adam Hughes, and Scott Beason. The collaboration with my friendly and brilliant colleagues greatly expanded my horizon. Especially, I want to thank Dr. Zhenhua Guo, for his invaluable and electrifying discussions with me on various research topics.

I would like to thank the administrative support from School of Informatics and Computing, FutureGrid, and University Information Technology Services. Because of their help, I am able to finish many research projects in time.

Finally, I want to thank my father Shanqing Ruan and my mother Xiaobin Guo, for their generous support during the past few years. I would not be able to finish the Doctor degree without their endless love and support.

Again, I am deeply grateful for all your support!

Thank you, all!

Abstract

During the past few decades, advances in the next generation of sequencing (NGS) techniques have enabled rapid analysis of the whole genetic information within a microbial community, bypassing the culturing of individual microbial species in the lab. These techniques have led to a proliferation of raw genomic data, which enables an unprecedented opportunity for data mining. To analyze a voluminous amount of bioinformatics data, a pipeline called DACIDR has been proposed. DACIDR adopts a taxonomy-independent approach to grouping these sequences into operational taxonomic units (OTUs), referred to as *data clustering*, and it enables visualization of the clustering result leveraging the power of parallelization and multidimensional scaling (MDS) techniques by utilizing large-scale computational resources. First, in order to observe the proximity of the sequences in a lower dimension, sequence alignment techniques are applied on each pair of sequences to generate similarity scores in a high dimension. These scores need to be assigned with weights in order to achieve an accurate result in MDS. Therefore, a robust and scalable MDS algorithm called WDA-SMACOF is proposed to address the issues of either missing distances or a non-trivial weight function. Second, the dataset with millions of sequences is usually divided into two parts: the first is processed with MDS, which has quadratic space and time complexity while the second is interpolated with approximation, resulting in a linear time complexity; this is also referred to as interpolation. In order to achieve real-time processing speed, a novel hierarchical approach has been proposed to further reduce the time complexity of interpolation to sub-linear. Thirdly, a phylogenetic tree is commonly used to demonstrate the phylogeny and evolutionary path of various organisms. A traditional way of visualizing phylogenetic tree preserves only the correlations between ancestors and their descendants. By utilizing MDS and interpolation, an algorithm called *interpolative joining* has been proposed to display the tree on the top of clustering, where their correlations can be intuitively observed in a 3D tree diagram called Spherical Phylogram. The optimizations in these three steps greatly reduce the time complexity of visualizing sequence clustering while increase its accuracy.

Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1. INTRODUCTION

1.1 Sequence Clustering and Visualization

Advances in modern bio-sequencing techniques have led to a proliferation of raw genomic data that need to be analyzed with various technologies such as pyrosequencing [1, 2]. These technology enables biologist generate mass gene sequence fragments within a short period of time. However, many existing methods lack efficiency on massive sequence collections analysis where the existing computational power on single machine can be overwhelmed. Consequently, new techniques and parallel computation must be brought to this area.

Existing techniques to analyze such data are divided into two categories: taxonomy-based and taxonomy-independent [3]. Taxonomy-based methods provide classification information about the organisms in a sample. For example, BLAST [4] relies on reference database that contains information about previous classified sequences, and compares new sequences against them, so that the new sequences can be assigned to the same organism with the best-matched reference sequence in the database. However, since most of the sequences are not formally classified yet, these methods cannot identify the corresponding organisms from these sequences. In contrast, taxonomy-independent methods use different sequence alignment techniques to generate pairwise distances between sequences, and then cluster them into operational taxonomic units (OTUs) by giving different threshold. Then by analyzing these sequences with some existing reference sequences, the evolutionary path of certain species could emerge. Many taxonomy-independent

methods were developed over past few years [5-7] . The key step among these methods is clustering, which is to group input sequences into different OTUs. However, most of these clustering methods require a quadratic space and time over the input sequence size. For example, hierarchical clustering is one of the most popular choices that have been widely used in many sequence analysis tools. It is a classic method, which is based on pairwise distance between input sequence samples. However, the main drawback of it is the quadratic space requirement for input distance matrix and a time complexity of $O(N^2)$. To overcome this shortage, several heuristic and hierarchical methods are developed and sometimes they can only perform on low dimensional data or lack accuracy [8, 9].

In order to visualize the clustering result from taxonomy-independent analysis in an intuitive way, dimension reduction techniques are used commonly in this field. It has been proved to be useful in data clustering and visualization field [5, 10]. This technique enables the investigation of unknown structures from high dimensional space into visualization in 2D or 3D space. Multidimensional Scaling [11] (MDS) is one set of techniques among many existing dimension reduction methods, such as Principal Component Analysis [12] (PCA) , Generative Topographic Mapping [13] (GTM), and Self-Organizing Maps [14] (SOM). Different from them, which focus on using the feature vector information in original dimension to construct a configuration in low dimension space, MDS focuses on using the proximity data, which is represented as pairwise dissimilarity values generated from high dimensional space. As in bioinformatics data, one needs to deal with sequences generated from sequencing technology, where the feature vectors are very difficult to be retrieved because of various sequence lengths. It is not suitable to use technologies other than MDS for their dimension reduction.

Another area of development is from the large scale computing, where the size of the clusters increases rapidly over the past few years [15]. And many innovative parallel computing frameworks has been proposed, to name a few, Hadoop [16], Spark [17], Pregel [18] and Twister [19]. In order to achieve large scale sequence analysis, these types of parallel computing framework must be used. So how to achieve maximum performance by leveraging the existing technologies remains a challenge.

This dissertation is mainly focused on optimizing MDS techniques and applies MDS technique into various situations in order to help biologist visualize the clustering result better. New algorithm has been proposed to improve the overall accuracy of MDS technique, and some optimization techniques has been proposed in order to lower the time cost as well. Finally, this dissertation also describes some detailed performance analyses and experiments related to the proposed methodologies.

1.2 Multidimensional Scaling

Multidimensional Scaling (MDS) is a set of statistic techniques used in dimension reduction. It is a general term for these techniques to apply on original high dimensional data and reduce their dimensions to target dimension space while preserving the correlations, which is usually Euclidean distance calculated from the original dimension space from the dataset, between each pair of data points as much as possible. This is a non-linear optimization problem in terms of reducing the difference between the mapping of original dimension space and target dimension space. In bioinformatics data visualization, each sequence in the original dataset is considered as a point in both original and target dimension space. The dissimilarity between each pair of sequences is considered as Euclidean distance used in MDS.

Given a data set of N points in original space, a pairwise distance matrix Δ can be given from these data points ($\Delta = [\delta_{ij}]$) where δ_{ij} is the dissimilarity between point i and point j in original dimension space which follows the rules: (1) Symmetric: $\delta_{ij} = \delta_{ji}$. (2) Positivity: $\delta_{ij} > 0$. (3) Zero Diagnosal: $\delta_{ii} = 0$. Given a target dimension L, the mapping of points in target dimension can be given by an $N \times L$ matrix X, where each point is denoted as x_i from original space is represented as ith row in X.

The object function represents the proximity data for MDS to construct lower dimension space is called STRESS or SSTRESS, which are given in equation (1) and (2):

$$\sigma(X) = \sum_{i<j\leq N} w_{ij} \left(d_{ij}(X) - \delta_{ij}\right)^2 \quad (1)$$

$$\sigma(X) = \sum_{i<j\leq N} w_{ij} \left(d_{ij}^2(X) - \delta_{ij}^2\right)^2 \quad (2)$$

where w_{ij} denotes the possible weight from each pair of points that $w_{ij} \geq 0, W = [w_{ij}]$, d_{ij} denotes the Euclidean distance between point i and j in target dimension.

It is easy to learn that the STRESS or SSTRESS [20] value actually represents the difference of between the distance calculated from the original dimension and the distance calculated from the mapping in the target dimension. The optimization process of MDS technique is used to minimize this difference. When the difference is minimized, the mapping in the target dimension will preserve most of the information among the data. As for the visualization from bioinformatics sequences, the sequence clustering sometimes will emerge naturally in the dimensionality reduction result (3D space), where one can easily observe the sequence clusters intuitively.

1.3 **Online MDS**

The traditional MDS problem are usually solved in $O(N^2)$ time and space. This is because of the requirement from the input of MDS that usually requires distances between all pairs of sequences. This makes MDS techniques hard to be applied on very large scale dataset. As the data size explodes during the past few years because of the next generation sequencing [12] (NGS) techniques, it is essential to find alternative method to address this problem. And the *in-sample* and *out-of-sample* solution has been brought up in data clustering and visualization to solve the large-scale data problem [21]. In this scenario, the whole dataset is divided into two parts: one part is called *in-sample* dataset, and the other part is called *out-of-sample* dataset. MDS is used to solve the *in-sample* problem, where a relatively smaller size of data is selected to construct a low dimension configuration space. And remaining *out-of-sample* data can be interpolated to this space without the usage of extra memory.

In formal definition, a dataset contains size of N sequences is divided into two parts: size of N_1 *in-sample* data, denoted as D_1, and size of N_2 *out-of-sample* points, denoted as D_2. The *in-sample* data are already mapped into an *L-dimension* space, and the *out-of-sample* data needs to be interpolated to an *L-dimension* space. These points in *L-dimension* is defined as $X=\{X_1,X_2\}$, where the *in-sample* points are $X_1 = \{x_1, x_2, x_3, \dots, x_{N_1}\}$ and the *out-of-sample* points are $X_2 = \{x_{N+1}, x_{N+2}, x_{N+3}, \dots, x_{N_2}\}$. Note that only one *out-of-sample* point at a time is interpolated to the

in-sample space. So the problem can be simplified to interpolate a point $\hat{x}$ to *L-dimension* with the distance observed to *in-sample* points. The STRESS function for $\hat{x}$ is given by

$$\sigma(X) = \sum_{i \le N} w_{i\hat{x}} \left(d_{i\hat{x}}(X) - \delta_{i\hat{x}}\right)^2 \quad (3)$$

where $d_i(\hat{x})$ is the distance from $\hat{x}$ to in-sample point i in target dimension, and $\delta_{i\hat{x}}$ is the original dissimilarity between $\hat{x}$ and point i. If all weights equals to 1, equation (3) is transformed to

$$\sigma(X) = \sum_{i \le N} \left(d_{i\hat{x}}(X) - \delta_{i\hat{x}}\right)^2 \quad (4)$$

As equation (4) is very similar to equation (1), similar optimization techniques could be apply to it and used to reduce the dimensionality of the sequences in the out-of-sample dataset. And out of sample data could be interpolated into the target dimension space one by one, and each out-of-sample data point is independent from each other as shown in Figure 1.1. Thus this method is also called online MDS. Majorizing Interpolative MDS [22] (MI-MDS) is an algorithm proposed to solve (4) where all weights equal 1.

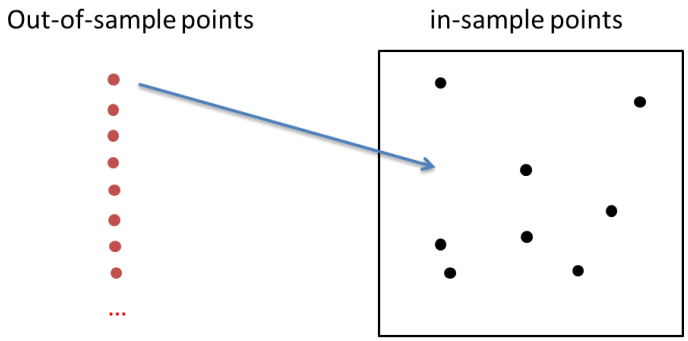

Figure 1.1: Illustrations of interpolate *out-of-sample* points into the *in-sample* target dimension space as 2D.

1.4 Phylogenetic Tree Visualization

Phylogenetic tree, or referred to as phylogeny is a general term for diagrams that illustrates the evolutionary relationships among different biological species, organisms or genes from a common ancestor [23]. These diagrams are usually shown in tree structure, where taxa joined together in the tree are implied to have descended from a common ancestor. Thus the name phylogenetic tree is called since the time of modern evolutionary theory. It is a very useful tool to consolidate

information of biological diversity and to provide perception into events that happened during evolution. Therefore, biologists tend to use phylogenies to visualize evolution, organize their knowledge of biodiversity, and guide ongoing evolutionary research.

Currently, phylogenetic tree can be displayed in various ways, which is reflected in the diversity of software tools available to biologists. Dendrogram [24], cladogram [25], phylogram and chronogram [26] are popular ways of displaying the phylogenetic tree. A dendrogram is a tree diagram originally used to illustrate the arrangement of the clusters produced by hierarchical clustering, but it is also used in computational biology to illustrate the clustering of genes or samples. A cladogram is formed using cladistics methods, which infer relations among organisms. However, it does not show the exact amount of change from ancestors to their related descendants. Therefore, this type of diagrams only represents a branching pattern. A phylogram is an evolutionary tree that has branch spans proportional to the amount of character change. So it has the same representations with a cladogram, except that the branch lengths vary from the ancestors to their descendants according to the amount of changes (either in time or genetic differences) between them. A chronogram is a phylogenetic tree that explicitly represents evolutionary time through its branch spans. In its representation, the branch lengths represent time, so current taxa are equidistant from the root. All of these display methods focused on displaying the relationships between the different taxa and their parent and ancestor, and it can help the biologist find the crucial evidence in an instructive way.

As increasing computing power enables researchers to construct ever-larger trees, displaying such large trees efficiently becomes a challenge. Constructing the diagrams in a low dimension space, 2D or 3D usually requires the knowledge of pairwise distances between each pair of taxa. Therefore, the online MDS technique could be used to reduce the dimensionality of the known species of the sequences. And this could be a solution to resolve the time complexity issue brought by large scale datasets.

1.5 Research Challenges

The particular characteristic of sequence clustering and visualization has brought many challenges. Firstly, the size of sequences generated every day is increasing rapidly, where utilizing the computing power of multiple machines (cluster) is essential as well as improving the efficiency of the algorithm. Secondly, the distances calculation between each pair of sequences does not behavior the same as in Euclidean distances. So the visualization algorithm should be optimized in order to use the distances in a correct way. Thirdly, visualizing a traditional phylogenetic tree and clustering separately is very inefficient for the biologist to observe the correlations between the results from separate algorithm. How to make the phylogenetic tree displaying method more efficient is still a challenge, and how to construct large scale phylogenetic tree can be difficult by using existing methods. So in order address these problems, this dissertation describes the solution from following areas:

1) Optimizing MDS algorithm

Applying weighting to the MDS algorithm is a nature method for handling distances generated from sequences, where unreliable distances can be set to a weight 0, and some significant distances could have a higher weight than 1. Furthermore, the weighting function will allow part of import sequences to be fixed with high weights, where rest sequences are varied according to their locations. The robustness is also required in MDS with weighting so that local optima can be avoided during the optimization process.

2) Reduce Time Cost for Online MDS

Although the online MDS has be brought to solve the time and space (memory) complexity issue for MDS, the time complexity for a single interpolated *out-of-sample* point remains to be high. As online algorithm requires the computation to be done within mille-seconds, the challenges remain for reducing the time complexity of online MDS. Although parallel computation power has been bought in, the algorithm itself still has room to improve.

3) Phylogenetic Tree Display with Clustering

Traditional phylogenetic tree display only shows the differences between taxa and their direct parent. The correlations between taxa are observed by using the path along branches. It is hard for the biologist to verify the clustering result from a separately generated phylogenetic tree. It is also a challenge to display large scale phylogenetic tree in an efficient way since organizing the branches with the known species is an action with high cost. It is even harder to observe the connections of the clustered sequences to the same sequences inside the phylogenetic tree if they are displayed separately when the number of sequences increases.

1.6 Contribution

By improving and utilizing MDS and online MDS algorithms, several algorithms have been proposed in this dissertation. The algorithms are listed in Table 1. The N is the total number of sequences, N_1 is the number of sequences in *in-sample* dataset, and N_2 is the number of sequences in *out-of-sample* dataset. The contribution of this dissertation is summarized as the following:

1) **Robust and Scalable MDS with Weighting**

By leveraging the power of conjugated gradient, the time complexity of MDS with weighting can be reduced from cubic to quadratic, which makes it suitable for large scale dataset. And Deterministic Annealing technique is applied to avoid the local optima from the original algorithm.

2) **Hierarchical Online MDS**

Use hierarchical algorithm instead of linear speed algorithm. Thus the time complexity can be reduced to logarithmic from linear. And weighting function is added to support the missing values from *in-sample* points.

3) **3D Phylogenetic Tree Visualization**

The 3D here means target dimension result after applying dimensionality reduction based on the pairwise distances between each pair of sequences. This result could be in 3D or 2D as well as it can be observed directly using naked eyes. The reason for 3D is that it is the highest dimension that can be seen by human. By constructing phylogenetic tree directly from dimension reduction result, the clustering result could be displayed directly with the tree since clusters are naturally

appeared during the MDS process. The interpolation algorithm could be used to find the coordinates for the internal nodes in the tree structure, so that cladogram or phylogram can be constructed.

Table 1 Summary of the algorithms proposed in this dissertation

	Time Complexity	**Space Complexity**	**Description**
Weighted DA-SMACOF (WDA-SMACOF)	$O(N^2)$	$O(N^2)$	Added weight function to DA-SMACOF
Fixed WDA-SMACOF	$O(N^2)$	$O(N^2)$	Fix part of points and varies other points
Weighted MI-MDS (W-MI-MDS)	$O(N_2N_1)$	$O(N_1)$	Added weight function to MI-MDS
Heurisitic MI-MDS (HE-MI)	$O(N_2 \log N_1)$	$O(N_1)$	Reduced the time cost of MI-MDS
Interpolative Joining (IJ)	$O(N_2N_1)$	$O(N_1)$	Generate the Spherical Phylogram

1.7 **Overview**

This dissertation is composed of several chapters, where each chapter covers a unique area of the research.

Chapter 2 mainly talks about the background techniques related to this thesis. First sequence alignment is discussed, followed by introduction of the hybrid parallel framework, MapReduce and iterative MapReduce used in this dissertation. The deterministic annealing technique is then described with an MDS algorithm called DA-SMACOF, followed by the introduction of phylogenetic analysis. Finally, a data clustering and visualization pipeline called DACIDR is described in detail, since all of the optimization techniques introduced in these dissertations were based on this pipeline.

Chapter 3 describes the WDA-SMACOF algorithm which enables scalable MDS for various situations with the distances. In this section, the equations and algorithm of WDA-SMACOF is described in detail, followed by the parallelization of WDA-SMACOF. Then a special case of WDA-SMACOF is discussed as Fixed-WDA-SMACOF algorithm. The experiments include the

comparison of WDA-SMACOF to other existing MDS algorithms in terms of both accuracy and time cost.

Chapter 4 introduces the W-MI-MDS algorithm and the hierarchical solution to reduce the time cost. At first how the weighting function added to MI-MDS is introduced. Then the parallelization of this algorithm is discussed, followed by the hierarchical method. In hierarchical method description, two tree structures noted as SSP-Tree and CN-Tree are discussed along with a heuristic method called HE-MI. The performance analysis of these methods includes the accuracy and time cost comparison with other methods, as well as the detailed analysis on the proposed algorithms.

Chapter 5 shows two new ways of displaying phylogenetic tree in 3D with clustering result. A method called Cuboid Cladogram is introduced first with the principle component analysis technique it was using for maximizing the variance of the visualized clusters. Then a more clearer method called Spherical Phylogram is described in detailed. It uses a new algorithm call Interpolative Joining that uses MDS and Interpolation techniques to construct the phylogenetic tree from clustering result. The experiments carried out in this section prove the effectiveness of these methods, as well as the importance of choosing a robust dimension reduction algorithm, such as WDA-SMACOF.

Chapter 6 is the conclusion and future work. The analysis result from the dissertation is discussed, then two methods involves future work is deliberated for possible improvement.

Chapter 2. A DATA CLUSTERING AND VISUALIZATION PIPELINE

2.1 Sequence Alignment

Biological similarity between two sequences is the property driving the sequence clustering and visualization. Thus, to form a measurable value of similarity we first align the two sequences and compute a distance value for each alignment, which represents the inverse of similarity and is used by algorithms down the line. A distance should be computed for each pair of sequences; hence the procedure is referred to as all-pair sequence alignment (ASA) or pairwise distance calculation (PDC).

The sequence alignment methods are divided into two categories according to their features: pairwise sequence alignment [5] (PWA) and multiple sequence alignment [27] (MSA). MSA is used for three or more sequences and it is usually more computationally complex than PWA. In many cases, the input set of query sequences are assumed to share a lineage and are descended from a common ancestor. So usually the sequences in MSA have an evolutionary relationship is commonly used in phylogenetic analysis. PWA aims to find an overlapping region of the given two sequences that has the highest similarity as computed by a score measure. The overlap may either be defined over the entire length or over a portion of the two sequences. The former is

known as global alignment and latter as local alignment. Needleman-Wunsch [28] (NW) and Smith-Waterman Gotoh [29, 30] (SWG) are two popular algorithms performing these alignments respectively. SWG performs local sequence alignment to determine similar regions between two nucleotide or protein sequence. It compares segments of all possible lengths and optimizes the similarity measure. In contrast, NW performs global sequence alignment trying to align every residue in every sequence.

Figure 2.1 shows a general sequence alignment with possible end gaps (note a local alignment will not result end gaps). We consider the region excluding end gaps as the aligned region. Pairs of boxes with the same color indicate a match and others indicate mismatches. Pairs with one box and one dash indicate a character being aligned with a gap. Two parameters governing NW and SWG are the scoring matrix and gap penalties, namely a gap open (GO) and a gap extension (GE) penalty.

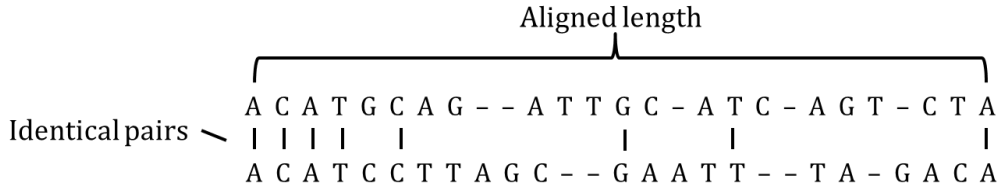

Figure 2.1: Illustration of calculating PID between two aligned sequences

Percentage identity is used to represent similarity among sequences, the distance between sequence *i* and sequence *j* is considered as the dissimilarity between them, as calculated in Equation (5):

$$\delta_{PID} = 1.0 - \frac{N_{ip}}{N_l} \quad (5)$$

where N_{ip} is the number of identical pairs between sequence *i* and sequence *j* and N_l is the aligned sequence length.

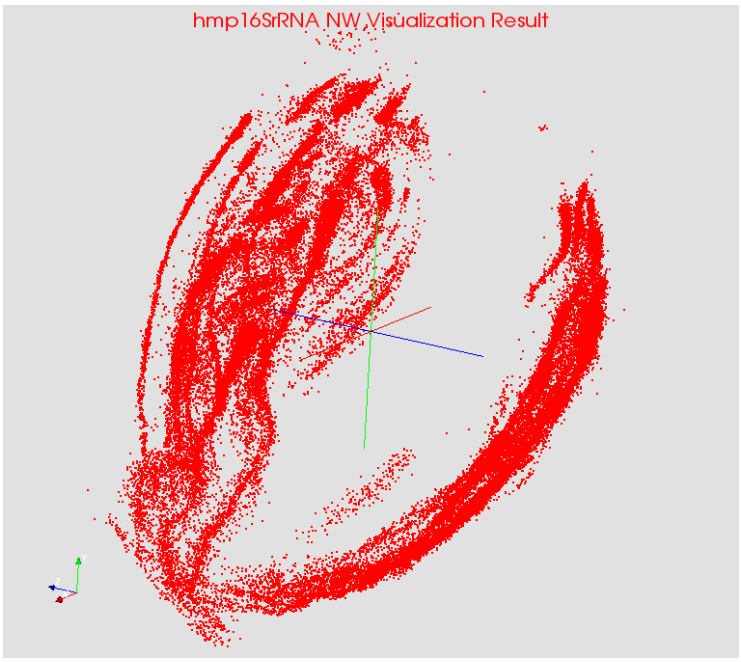

Figure 2.2: Visualization of 16S rRNA data with NW pairwise sequence alignment result

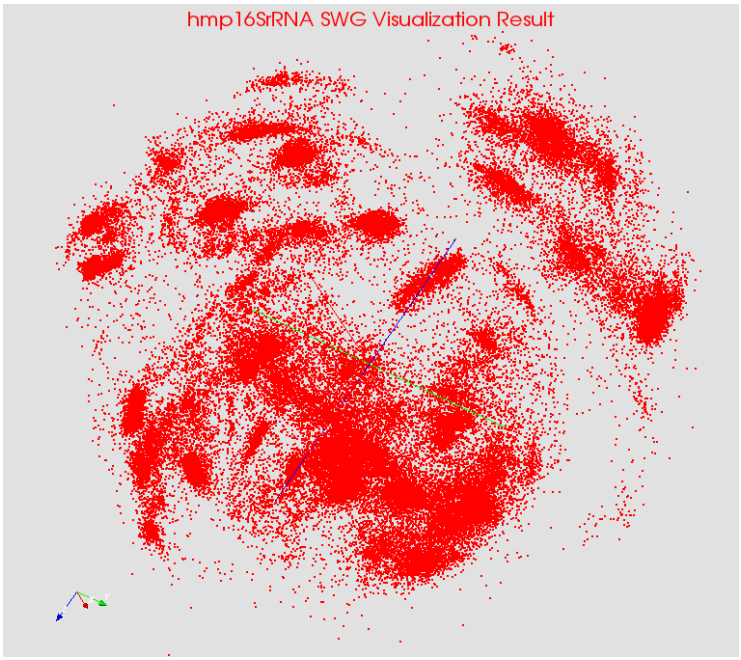

Figure 2.3: Visualization of 16S rRNA with SWG pairwise sequence alignment result

Generally speaking, if the sequences in the dataset are similar and of roughly equal size, NW is more preferred than SWG. And for the sequences with various lengths, SWG can generate a more accurate result than NW. Figure 2 shows an example of the visualization result of a same sequence

dataset using SWG and NW. The SWG obviously produces more reliable results than NW because of the clearer clusters visualized. This is because the sequence lengths were not uniform in this particular hmp16S rRNA dataset. And NW, being a global alignment algorithm, had done its best by producing alignments with many gaps. In cases where a shorter sequence is aligned with a longer one, the gaps were dearly added by NW simply to make the alignment from end to end. Unfortunately, the distance measure used to compute over the alignments was susceptible to gaps and produced artificially large distances for sequence pairs. The plots we generated with NW based distances had long thin cylindrical point formations as shown in Figure 2.2, which later is identified as a direct consequence of the number of gaps present in the alignment. Pictorially, this effect is shown in Figure 2.3. From the DACIDR result, multiple points selected on the same cylinder belong to a same cluster, but by using NW, instead of clustered, these points are aligned in line. The selected points are based on their ID number in the given sample dataset, where their lengths are 507 to 284.

PWA algorithm is time consuming, and for all-pair problem, the time and space complexity is $O(N^2)$. Thus, it is not practical to run millions of sequence alignments using on a single machine. However, ASA is an embarrassingly parallel problem and thus we have mapped it into MapReduce paradigm by adopting coarse granularity task decomposition. The parallelized ASA makes it possible to generate large dissimilarity matrices resulting from aligning millions of sequences and has been proved to be highly efficient in our previous work. The way of parallelizing it is shown in Figure 2.4. The target distance matrix is divided into blocks, so that each processor will process only part of the matrix. The blocks in the darker color are the blocks that need to be calculated, because of the symmetric property for a Euclidean distance matrix, i.e. Block(i, j) = Block(j, i). And in practice, the sequence alignment from sequence A to sequence B is very similar to sequence alignment from sequence B to sequence A. So these two alignments will be considered as same. And the reason for dividing the workload this way is for load balancing, so that each processor can process a set of blocks based on the block row id or column id.

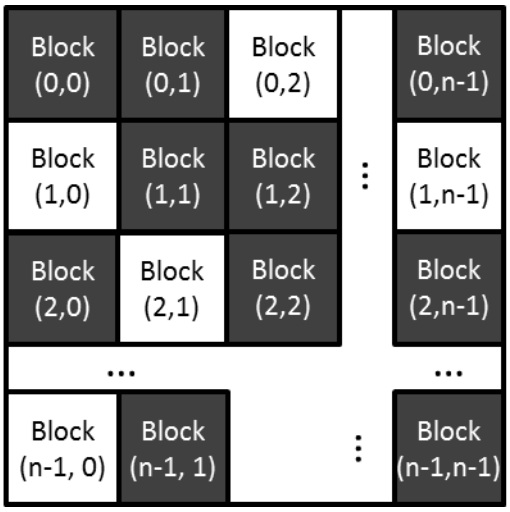

Figure 2.4: Parallelization of the ASA problem. The total number of sequence is n, and the darker block is the block needs to be processed, and white blocks are their symmetric blocks.

2.2 Hybrid MapReduce Workflow

MapReduce is a parallel programming model proposed by Google to support large-scale data processing [31]. Hadoop is an open source implementation of MapReduce with a distributed file system (HDFS) [32]. Each MapReduce job takes a set of key/value pairs as input, and produces a set of key/value pairs. The computation of MapReduce jobs is split into 2 phases: map and reduce. In map phase, map function takes an input key/value pair, read the data accordingly from HDFS, and produces a set of intermediate key/value pairs. Hadoop groups together all intermediate values associated with the same intermediate key and passes them to the reduce function. In reduce phase, each reduce operation accepts an intermediate key and all values associate with that key. It merges these values to form a possibly smaller set of values, emits key/value pairs of the final output, and writes the final output to HDFS.

Hadoop adopts master-slave architecture. The master node runs a namenode, a secondary namenode and a job tracker. The namenode is mainly used for HDFS for hosting the file system index; the secondary namenode can generate snapshots of the namenode's memory structures, thus preventing file system corruption and reducing loss of data; the job tracker allocates work to the task tracker nearest to the data with an available slot. The slave node runs a datanode and a task tracker: datanode contains blocks of data inside HDFS where multiple datanodes can serve up

distributed data over network; task tracker will spawn java processes as workers to execute the work received from job tracker. Hadoop supports fault tolerance in both MapReduce execution and HDFS. HDFS supports fault tolerance by providing file replicas among the slave nodes. In case one or several datanodes fail, the integrity of the distributed files won't be harmed by using the replicas from other running datanodes. During a MapReduce job execution, once a task tracker fails, all the unfinished tasks on that task tracker will be scheduled to the empty slots of other task tracker.

Many data analysis applications require iterative computations, such as deterministic annealing pairwise clustering [33] and dimension reduction [34] algorithms. This type of applications can be parallelized with MapReduce paradigm. However, they have a unique feature that is to keep running map and reduce iteratively until the computation satisfies a condition to converge to a final result.

Hadoop has been proved to be useful in large scale data parallel distributed computing job. However, it does not directly support iterative data analysis applications. Instead, the iterations must be orchestrated manually using a driver program where each iteration is piped as a separate MapReduce job. There are several problems: Firstly, as the iteration number is manually set by the user, it is impossible to make the program converge to meet a certain condition; Secondly, the static data needs to be load from disk to memory in every iteration which can generate high network I/O and disk I/O overhead; Thirdly, the job tracker needs to reschedule map and reduce tasks for every iteration, which brings considerable scheduling overhead.

Therefore, the iterative applications are parallelized using Twister [19], an iterative MapReduce framework. Twister uses pub/sub messaging for all the communication/data transfer where a broker will be running on one of the compute nodes during the execution. All the other nodes are classified as compute nodes where the Twister daemon runs on. Twister daemon can connect to broker and spawn threads executing map and reduce tasks. The client driver of Twister is used to support iterations of map and reduce tasks. The map and reduce tasks won't exit after one iteration unless the client driver sends the exit signal. So an iterative MapReduce job is not considered as

multiple MapReduce jobs as in Hadoop. Twister supports for long running mappers and reducers with an in memory data model. This design eliminates the overhead of data reloading from disk to memory across iteration boundaries. Twister schedules map and reduce tasks before the first iteration so that they are processed by the same mappers and reducers during each iteration for locality. This can eliminate the scheduling overhead of task rescheduling. In summary, Twister is optimized for iterative parallel applications, where Hadoop doesn't perform well. Even though Twister runs fast for iterative parallel applications, current implementation of Twister lacks several features for large-scale usage: distributed file system support, dynamic scheduling and fault tolerance. So a hybrid MapReduce workflow management system is developed to support the fast execution of iteration applications such as MDS, as well as the reliable execution for applications such as ASA.

2.3 Deterministic Annealing

Deterministic annealing [35] (DA) is an annealing process that finds global optima of an optimization process instead of local optima by adding a computational temperature to the target object function. By lowering the temperature during the annealing process, the problem space gradually reveals to the original object function. Different from Simulated Annealing [36], which is based on Metropolis algorithm for atomic simulations, it neither rely on the random sampling process nor random decisions based on current state. DA uses an effective energy function, which is derived through expectation and is deterministically optimized at successively reduced temperatures.

Scaling by Majorizing a Complicated Function [37] (SMACOF) is a STRESS majorization algorithm solving the MDS problem. The object function is set as the STRESS function in equation (1). And by setting an upper bound of this convex function, a majorization formula could be derived. By iteratively setting the estimated dimension reduction result as the initial conditions in that formula, the algorithm can find the mapping of the target dimension with a non-increasing STRESS value iteratively. However, since this algorithm is an EM-like [38] algorithm, this

mapping result could be trapped under local optima. DA-SMACOF [39] is proposed to solve this issue with DA optimization.

The goal of DA in SMACOF is to minimize $\mathcal{F}_{\mathrm{mds}}(\mathrm{P}_0) =< \mathcal{H}_{\mathrm{mds}} - \mathcal{H}_0 >_0 + \mathcal{F}_0(\mathcal{P}_0)$ with respect to parameters $\mathrm{u_i}$ is independent of $< \mathcal{H}_0 >_0 + \mathcal{F}_0(\mathcal{P}_0)$, so the problem can be simplified to minimize $< \mathcal{H}_{\mathrm{mds}} >$ if we ignore the terms independent of $\mathrm{u_i}$. By differentiating (1), we can get

$$< (\mathrm{x_i} - \mathrm{x_j})^2 >= (\mathrm{u_i} - \mathrm{u_j})^2 + 2\mathrm{TL} \quad (6)$$

where $\mathrm{x_i}$ is the ith point in the target dimension L, as same as ith line in matrix X.

Take (6) into (1), finally the $\mathcal{H}_{\mathrm{MDS}}$ became

$$\mathcal{H}_{\mathrm{MDS}} = \sum_{\mathrm{i<j\leq N}} \mathrm{w_{ij}} \left(\sqrt{(\mathrm{u_i} - \mathrm{u_j})^2 + 2\mathrm{TL}} - \delta_{\mathrm{ij}}\right)^2 \quad (7)$$

$$= \sum_{\mathrm{i<j\leq N}} \mathrm{w_{ij}} \left(\left|\mathrm{u_i} - \mathrm{u_j}\right| + \sqrt{2\mathrm{TL}} - \delta_{\mathrm{ij}}\right)^2 \quad (8)$$

As the original cost function and target dimension configuration gradually changes when the computational temperature changes, we denote $\mathrm{X_T}$ as the target dimensional configuration and Δ_{T} as the dissimilarities of each pair of sequences under temperature T. So the updated STRESS function of DA-SMACOF becomes

$$\sigma(\mathrm{X_T}) = \sum_{\mathrm{i<j\leq N}} \mathrm{w_{ij}} \left(\mathrm{d_{ij}}(\mathrm{X_T}) - \tilde{\delta}_{\mathrm{ij}}\right)^2 \quad (9)$$

where $\tilde{\delta}_{\mathrm{ij}}$ is defined as

$$\tilde{\delta}_{\mathrm{ij}} = \begin{cases} \delta_{\mathrm{ij}} & \text{if } \mathrm{w_{ij}} = 0 \\ \delta_{\mathrm{ij}} - \sqrt{2\mathrm{TL}} & \text{else if } \delta_{\mathrm{ij}} > \sqrt{2\mathrm{TL}} \\ 0 & \text{other wise} \end{cases} \quad (10)$$

Note that if the distance between point i and point j is missing from Δ, then $\mathrm{w_{ij}} = 0$. There is no difference between $\tilde{\delta}_{\mathrm{ij}}$and δ_{ij} since both of the distances are considered missing values. This is not proposed in the original DA-SMACOF where all weights for all distances in Δ are set to 1.

And the final formula of the DA-SMAOCF can be derived as the following:

$$\mathrm{VX_T} = \mathrm{B}(\mathrm{Z_T})\mathrm{Z_T} \quad (11)$$

$$X_T^u = V^{\dagger} B(Z_T) Z_T \quad (12)$$

where $V^{\dagger}$ is the pseudo-inverse of V. T means the current temperature, and Z is the estimated X from previous iteration. Equation (12) is also called Guttman transform [37].

The V and $B(X)$ from equation (12) is defined as following:

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ -\sum_{k \neq i} v_{ik} & \text{else if } i = j \end{cases} \quad (13)$$

$$b_{ij} = \begin{cases} -\frac{w_{ij}\tilde{\delta}_{ij}}{d_{ij}(X_T)} & \text{if } i \neq j \text{ and } d_{ij}(X_T) \neq 0 \\ -\sum_{k \neq i} b_{ik} & \text{else if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

Note that in DA-SMACOF, all weights equal one, so in equation (14), the weights have no differences being associate with each pair of distances.

2.4 Phylogenetic Analysis

Phylogenetic analysis is usually targeted to study the evolutionary relationships among the different species or genes. As the modern sequencing technologies in molecular biology advances rapidly these days, large amount of sequence data are collected from different organisms. In order to study the relatedness among these sequences, evolutionary studies using phylogenetic analysis is still of very high interest. These studies mainly focused on the morphology of the organism. By doing analysis on the sequences, one can classify an organism by employing a system of record keeping its characteristics as well as the diverse sets of comparative information. Naturally the tree structure is commonly used to serve this purpose [40].

Phylogenetic trees are tree representation that constructed to record the classifications of organisms. It is a very informative and intuitive way to describe the evolutionary path of certain group of organisms. Since the evolution is a branching process, the tree representation is naturally a best representation to describe the procedure. The children split from parent can be view as populations diverged over time, or terminated by extinction. There are many popular tools or methods for inferring phylogenies, to name a few, such as maximum likelihood [41], neighbor joining [42], Monte Carlo approach [43] and Bayesian inference [44] techniques. Among them, a

software called RAxML [45] is the expectation maximization based software, whereby it has been proved to have a very high reliability during the phylogenetic analysis. Usually one has to do MSA before applying the RAxML on the target sequence set in order to generate a phylogenetic tree with the capability of displaying phylogram.

Besides construction of the phylogenetic tree, the other key aspect of phylogenetic analysis is the display of the tree. As mentioned previously, most of the displaying methods only contain the information between the organisms and their ancestors, but not the correlations between the organisms. Some most common displaying methods, such as rectangular cladogram and phylogram are shown in.

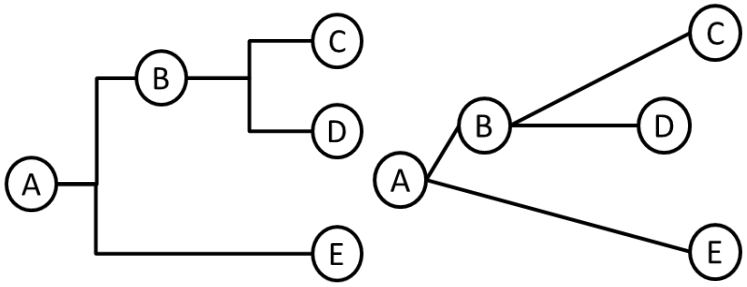

Figure 2.5: 2D tree diagram example for 5 sequences, left one is the rectangular cladogram, and right one is the rectangular phylogram

In these representation, note that the branches in the cladogram always have the same lengths, and the branches in the phylogram will have different branch lengths which represents the amount of changes from a parent to its child. Assuming that both graph were drawn based on a same phylogenetic tree, where A and B are the internal node and C, D, E are the terminal node (leaf node). C, D, E are usually representing OTUs, and A, B are the ancestors without empirical data, so they are representing Hypothetical Taxonomic Units (HTUs). Both of these diagrams are rooted tree examples, where they share a common ancestor as A.

2.5 A Clustering and Visualization Pipeline

We proposed a method pipeline called deterministic annealing clustering and interpolative dimension reduction (DACIDR) [46] that can be collectively classified as taxonomy-independent analysis, wherein different sequence alignment tools are applied in order to glean specific pieces

of information about the related genome. This pipeline combines the techniques from previous sections.

We used deterministic annealing method for dimension reduction and pairwise clustering to group the sequences into different clusters and visualize them in a lower dimension. In DACIDR, the input dataset is divided into in-sample dataset and out-of-sample dataset. The *in-sample* set is processed using ASA, PWC and MDS, while *out-of-sample* set is processed by Interpolation. In detail, as shown in the top part of Figure 2.6, DACIDR includes all-pair sequence alignment (ASA), pairwise clustering (PWC), multidimensional scaling (MDS), interpolation and visualization. The ASA reads a FASTA file and generates a dissimilarity matrix; The PWC can read the dissimilarity matrix and generate OTUs; MDS reads dissimilarity matrix and generates a 3D mapping; Interpolation (online MDS) read the OTUs and plots to generate mapping for further sequences.

The DACIDR can manage to generate the clustering and visualization result only with the input of sequences, and for very large dataset, i.e. millions of sequences, the clustering result is also referred as mega regions, as inside each region, there are hundreds of thousands of sequences. These mega regions still have internal structures which seem to be several sub-clusters. These sub clusters on a plot with the whole dataset couldn't be shown clearly because the distance between regions are relatively larger than the distance between sub-clusters in each region. So the points in each region are tend to be closer to each other, thus the differences are diminished. In order to get a finer resolution of the clusters, DACIDR can then be applied on each of the region separately. This recursive process is also referred as recursive clustering as shown in the middle part of Figure 2.6. This process can be done as many times as a target granularity of clusters is reached.

After the clusters are found with a satisfied resolution, the center of each cluster could be calculated using the target dimension points corresponding to the input sequences. This is usually done because of the need for phylogenetic analysis. As a phylogenetic tree with over millions of sequences would hard to understand, each cluster is only represented by a sequence which is the closed to the center of a cluster. So that the number of sequences displayed in a phylogenetic tree

could be dramatically reduced. By adding another reference sequence dataset from some well-defined phylogenetic trees [47] or GenBank [48], a traditional phylogenetic tree could be construct with some well-known methods such as RAxML. The same DACIDR pipeline could be apply on this dataset again considering all the sequences as in-sample dataset to generate a visualization result in 2D or 3D space (3D is usually chosen because it can retain more information during the dimension reduction). Note that by just doing the proposed MDS algorithm, the clusters will naturally appear in the 3D space. A proposed algorithm called Interpolative Joining can determine this phylogenetic tree in 3D space by combining the result from DACIDR and RAxML, so that a final spherical phylogram could be generated. This novel approach will allow the biologist to observe the correlations between clustering result and phylogenetic together. The details will be described in section 5.

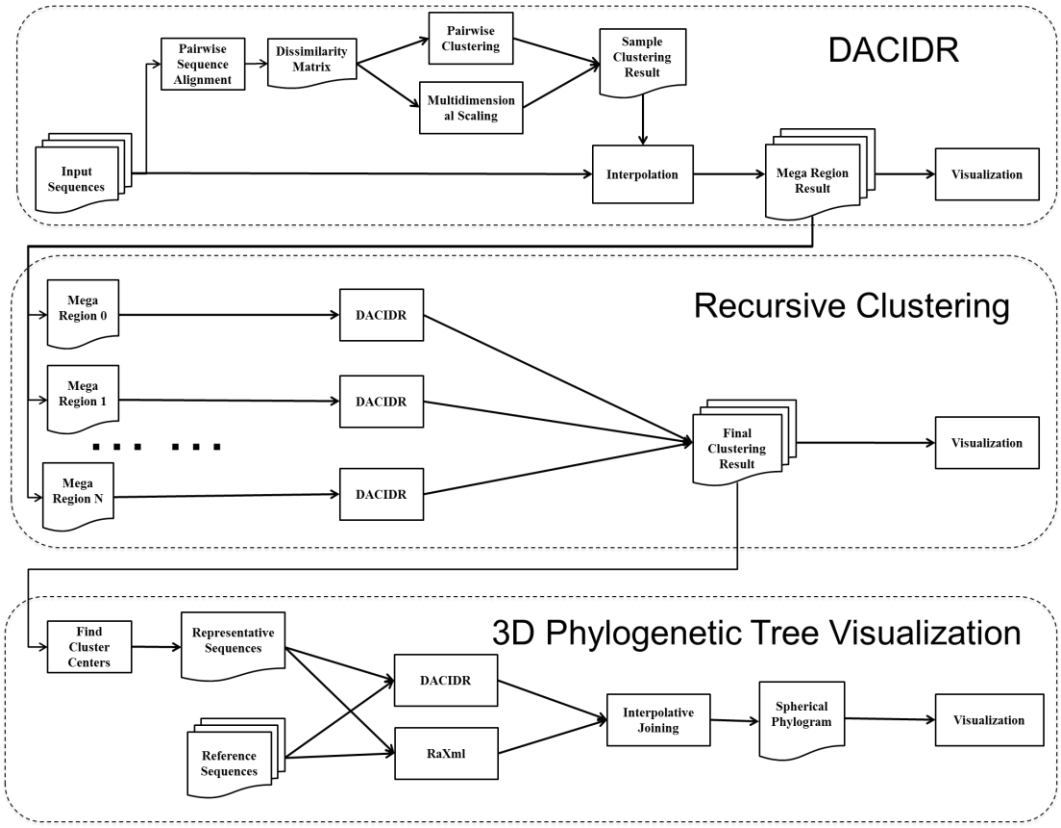

Figure 2.6: The flowchart of DACIDR of processing over millions of sequences until the phylogenetic tree is visualized based on previous experience.

Note that all of these techniques are parallelized to process large data on multiple compute nodes, using MapReduce, iterative MapReduce and/or MPI [49] frameworks using the hybrid MapReduce workflow management system [50] [51].

Chapter 3. ROBUST AND SCALABLE MULTIDIMENSIONAL SCALING WITH WEIGHTING

3.1 Overview

MDS is heavily used in DACIDR in order to generate the visualization result of input sequences. As mentioned before, the sequence alignment, SWG sequence alignment performs better than NW sequence alignment where sequence lengths vary. However, SWG sometimes could generate very short sequence alignment, suggests that the sequence alignment has a lower quality compared to other longer alignments. This could potentially leads to in-accurate target dimension mapping. Furthermore, fixing reference sequence dataset and varying other sequences is an interesting topic. This is because reference sequences are usually well defined and have very clear structure after dimension reduction. One could compare unidentified sequences to the reference sequences in order to classify those sequences. As reference sequences set usually has a relative small size compared to the number of unidentified sequences, the structure of those sequences could be lost if they are simply interpolated. One way of discovering the relationships between unidentified sequences with reference sequences is to fix the position of reference sequences in the target

dimension space, and varies the unidentified sequences. This would also require a reliable MDS algorithm. Last but not least, the parallelization of these types of algorithm remains a challenge. As all MDS algorithms require $O(N^2)$ memory, so how to parallelize that efficiently in order to process as large size of data as possible with limited resources is non-trivial. And in this section, the solutions to the problems proposed here are presented. Section 3.2 mainly talk about related work in this field, Section 3.3 describe the WDA-SMACOF algorithm, which is the algorithm solves the time complexity issue associate with weighting function; the parallelization of WDA-SMACOF is presented in Section 3.4, followed by performance analysis in Section 3.5. And finally, the conclusion is given in Section 3.6.

3.2 **Related Work**

Many MDS algorithms have been proposed in the past few decades, and these techniques has been divided into metric multidimensional scaling and non-metric multidimensional scaling. Metric multidimensional scaling requires the distances to be Euclidean distances, where the symmetric property and triangular inequality property must be met within the distances in the original high dimensional space. Since the dissimilarities calculated during all-pair sequence alignment do not necessarily fit all these requirements, non-metric multidimensional scaling is used to solve the problem. In non-metric MDS, the Euclidean distances can be replaced with dissimilarities, where these properties do not needs to be hold as long as the STRESS or SSTRESS value in equation (1) and (2) is reduced during the optimization process. Newton's method [20] is used as a solution to minimize the STRESS in (1) and SSTRESS in (2). It is a gradient descent type of algorithm where the STRESS function is considered as the cost function. And in each iteration, the derivative of matrix X is taken. This method used the Hessian to form a basic Newton iteration, and then iterated through it until convergence. Although the time complexity of its conversion is quadratic, both Hessian construction and inversion require cubic time complexity. Quasi-Newton [52] method is proposed to solve this problem by using an approximation of inverse Hessian at each iteration. This significantly reduced the time complexity of Newton method to sub-cubic. Although the time complexity from this method is reduced compared to original Newton method,

it is still much higher than the conversion time. An Multi-Grid MDS (MG-MDS) [53] has been proposed to solve the isometric embedding problems. As a parallel solution, it shows the dramatic increase in performance compared to other existing methods. Scaling by Majorizing a Complicated Object Function (SMACOF) [11] is an majorization algorithm which is widely used for large-scale MDS problems. However, it involves full matrix inversion before the calculation with weighting, which always has cubic time complexity. Additionally, as this method is an Expectation Maximization (EM) like problem, it is suffered from local optima problem. So a DA solution has been added to SMACOF, so called DA-SMACOF [39], where it increased mapping quality and decreased the sensitivity with respect to initial configuration. Simulated Annealing [36] and Genetic Algorithm [54] have also been used to avoid the local optima in MDS. However, they suffered from long running time due to their Monte Carlo approach.

3.3 Weighted Determnistic Annealing SMACOF

3.3.1 Algorithm Description

As mentioned in Section 2.4, the DA-SMACOF used DA technique in order to avoid the local optima found in SMACOF. The drawback of DA-SMACOF is that when it assumes that all weights equal 1 for all distances, so the equation DA-SMACOF trying to solve is:

$$\sigma(X) = \sum_{i<j\leq N}(d_{ij}(X) - \delta_{ij})^2 \quad (15)$$

And the final formula in SMACOF is equivalent to:

$$\mathrm{X} = \frac{1}{N}\mathrm{B(Z)Z} \quad (16)$$

where N is the number of points (rows) in X and Z is the previously estimated X in last iteration.

It is intuitively to see that this equation requires quadratic time to solve since the matrix multiplication of B(Z), which is a N×N matrix and Z, which is N×L matrix has a time complexity of O(N×N×L). When L (usually 2 or 3) is much smaller than N, this algorithm has a time complexity of $O(N^2)$. And DA-SMACOF is adding a temperature to it, so that delta varies under different temperature, more iterations are added to the algorithm but the time complexity remains the same.

The problem comes when different weights are needed for this algorithm, since it states in equation (12) that a matrix inversion for an order N matrix V is needed, the time complexity of it is cubic. When N is large enough, i.e. hundreds of thousands, the time complexity of this algorithm becomes $O(N^3)$ despite the quadratic convergence speed for SMACOF. Although $V^{\dagger}$ could be calculated separately from SMACOF algorithm since V is static during the iterations, the time complexity of full rank matrix inversion is always $O(N^3)$ [55, 56]. Compared to the time complexity of SMACOF, which is $O(N^2)$, this is bottleneck for large-scale computation of weighted SMACOF.

Instead of using pseudo-inverse of V, WDA-SMACOF uses CG to solve the matrix inversion problem [57]. First, we denote $V + I$ as $\dot{V}$ and if N is large, $V + I \approx V$, where I is an $N \times N$ identity matrix, so by replacing V by $\dot{V}$ in (12), we have the majorizing function of WDA-SMACOF as

$$\dot{V}X_T = B(Z_T)Z_T \qquad (17)$$

Theorem 1. $\dot{V}$ is a symmetric positive definite (SPD) matrix.

Proof. Since $w_{ij} = w_{ji}$, so $v_{ij} = v_{ji}$, and $\dot{V} = \dot{V}^T$. From (13), $\dot{V}$ can be represented as

$$\dot{v}_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ 1 + \sum_{k \neq i} w_{ik} & \text{else if } i = j \end{cases} \qquad (18)$$

Because $w_{ij} \geq 0$, so $\dot{v}_{ii} > 0$. And $\dot{v}_{ii} > \sum_{k \neq i} w_{ik} = \sum_{k \neq i} |\dot{v}_{ik}|$. So according to[58], **Theorem 1** is proved.

Since $\dot{V}$ is an SPD matrix, we could solve (17) instead of (12) without doing the pseudo-inverse of V. And since equation (17) is in the form of $Ax = b$, a well-known iterative approximation method, so called Conjugate Gradient [59] (CG) could be used here to address this issue. CG is a gradient descent type of optimization technique as an addition to the steepest descent [60] method. Traditionally, it is used to solve quadratic form while x and b are both vectors. Although theoretically CG only converges when there is N iterations for an order N matrix, the iteration are much less needed in practice.

In our case, B(Z)Z and X are both $N \times L$ matrices. So the original CG could be directly used when $L = 1$. Nevertheless, for $L > 1$ situations, the CG method needs to be updated using following

equations. In ith iteration of CG, the residual is denoted as r_i, the search direction is denoted as d_i, α_i and β_i are scalars which represents the amount of directions needs to be updated with for X_i and for d_i. So r_0 and d_0 are given as

$$r_0 = d_0 = \dot{B} - \dot{V}X \quad (19)$$

where $\dot{B}$ is the produce of $B(Z) \times Z$.

Let's denote $\mathrm{dot}(X, Y) = \sum_{1 \le j \le L} \sum_{1 \le i \le N} x_{ji} y_{ij}$ where X is $L \times N$ and Y is $N \times L$ matrix and x_{ji} is the jth row, ith column element in X and y_{ij} is the ith row, jth column element in Y. In another word, $\mathrm{dot}(X, Y)$ is calculating the sum of dot product over rows of X and their corresponding columns in Y. So the complete equations for CG are updated to

$$\alpha_i = \frac{\mathrm{dot}(r_i^t, r_i)}{\mathrm{dot}(d_i^t, \dot{V}d_i)} \quad (20)$$

$$X_{i+1} = X_i + \alpha_i d_i \quad (21)$$

$$r_{i+1} = r_i - \alpha_i \dot{V} d_i \quad (22)$$

$$\beta_{i+1} = \frac{\mathrm{dot}(r_{i+1}^t, r_{i+1})}{\mathrm{dot}(r_i^t, r_i)} \quad (23)$$

$$d_{i+1} = r_{i+1} + \beta_{i+1} d_i \quad (24)$$

Denote the iterations in the SMACOF algorithm as SMACOF iteration and iterations in CG algorithm as CG iteration. Note that in each SMACOF iteration, only one matrix multiplication B(Z)Z with time complexity of $O(N^2)$ is needed. And it is a recognized fact that original CG is an iterative algorithm, that X and the other parameters are updated in each iteration. And the residual *r* is a non-increasing value until converge. So the time complexity of CG is $O(N^2)$ as the matrix multiplication in (20) and (22) are $N \times N \times L$ where $L \ll N$. And for one SMACOF iteration, multiple CG iterations are needed for the approximations of X. If averagely there are n_c iterations per SMACOF iteration, and n_s iterations for the WDA-SMACOF algorithm to converge, the time complexity of this algorithm would be $O(N*N*L*n_c*n_s)$. If n_c is small enough, this algorithm has a time complexity of $O(N^2)$.

Algorithm 1 WDA-SMACOF algorithm

Input: Δ,**W,** $\boldsymbol{X}$**,** $\boldsymbol{\varepsilon}$ **and** $\boldsymbol{\alpha}$
Output: X as target dimension mapping
1: Generate random initial mapping $\boldsymbol{X}$.
2: $\boldsymbol{k} = \boldsymbol{0}$;
3: while $\boldsymbol{T_k} \geq \boldsymbol{T_{min}}$ do
4: Compute $\boldsymbol{T_k}$ and $\tilde{\Delta}_{\boldsymbol{k}}$ using (10).
5: $\boldsymbol{t} = \boldsymbol{0}$;
6: while $\boldsymbol{\sigma}(\boldsymbol{X}^{|\boldsymbol{t}|}) - \boldsymbol{\sigma}(\boldsymbol{X}^{|\boldsymbol{t+1}|}) > \boldsymbol{\varepsilon}$
7: Use CG defined from (20) to (24) to solve (17).
8: $\boldsymbol{t} = \boldsymbol{t} + \boldsymbol{1}$;
9: end while
10: Cool down computational temperature $\boldsymbol{T_{k+1}} = \boldsymbol{\alpha}\, \boldsymbol{T_k}$;
11: $\boldsymbol{k} = \boldsymbol{k} + \boldsymbol{1}$
12: end while
13: $\boldsymbol{X}$ =output of SMACOF based on $\boldsymbol{T_k}$
14: return $\boldsymbol{X}$

Finally, the WDA-SMACOF algorithm is given in algorithm 1. WDA-SMACOF algorithm is illustrated in Algorithm 1. The initial temperature T_0 is critical in WDA-SMACOF that a flat initial configuration (all distance in Δ_{T_0} equals to zero) needs to be avoided. So the T_0 is calculated based on maximum value of weight times distance. The T_{min} is a number close to zero that can gives reasonable amount of temperature deduction. Overall, as mentioned before, the original SMACOF uses an $O(N^3)$ matrix inversion first, then do an $O(N^2)$ matrix multiplication in each iteration. WDA-SMACOF does the same $O(N^2)$ matrix multiplication, and one CG approximation in each SMACOF iteration as well. Therefore, WDA-SMACOF has a much higher scalability than original SMACOF proposed as Guttman transform.

3.3.2 **Parallel WDA-SMACOF**

The input data of WDA-SMACOF has three parts, the distance matrix Δ, the matrix $\dot{V}$ and the matrix W. Note that different from DA-SMACOF, the weight matrix W, and $\dot{V}$ are included during the computation, so the memory usage of WDA-SMAOCF is higher compared to DA-SMAOCF. However, since both W and $\dot{V}$ are $N \times N$ matrices, WDA-SMACOF still has memory (space) complexity of $O(N^2)$. In order to solve the memory problem, only W is store in the static memory during the iterations since each element of $\dot{V}$ is simply the negative value of the corresponding value in W except the diagnose elements. So only a one dimensional array with length N is needed

for $\dot{V}$ to store the diagnose elements, and all other elements are read from W directly when $\dot{V}$ is needed.

The parallelized WDA-SMACOF uses three single MapReduce computations and one nested iterative MapReduce computation for CG computation in one SMACOF iteration as outer loop. The main driver handles the starting and ending of each MapReduce job and some calculations that does not needs to be parallelized. As shown in equation (17), these three single MapReduce job are for calculation of function B(Z), calculation of matrix multiplication of B(Z)Z and one STRESS calculation in equation (1). The nested iterative MapReduce computation has one single MapReduce computation in one CG iteration as inner loop. Note that in the CG calculation, $\dot{V}d_i$ can be reused in equation (20) and in equation (22). And since it has a relatively small size (N ×L), it can be stored inside the memory of the main driver. So only one matrix multiplication needs to be parallelized in CG computation. In general, the computations in outer loop contain two matrix multiplication and one STRESS calculation. The computation in inner loop performs approximation in CG, as illustrated in Figure 3.1. Only the matrix X is needed to be synchronized on each mapper in both SMACOF iteration and CG iteration, so that the largest parameter broadcasted in each iteration is an $N \times L$ matrix, where L is often set to 2 or 3 for visualization purpose. So the communication overhead is low compared to the computation on each mapper in this design. And the memory has been reduced to half of the original design because of the elimination of storing whole V matrix inside memory. Finally, when the main driver decides this algorithm has converged with the condition $\sigma(X^{|t|}) - \sigma(X^{|t+1|}) > \varepsilon$, the whole computation will stop and yields the final result.

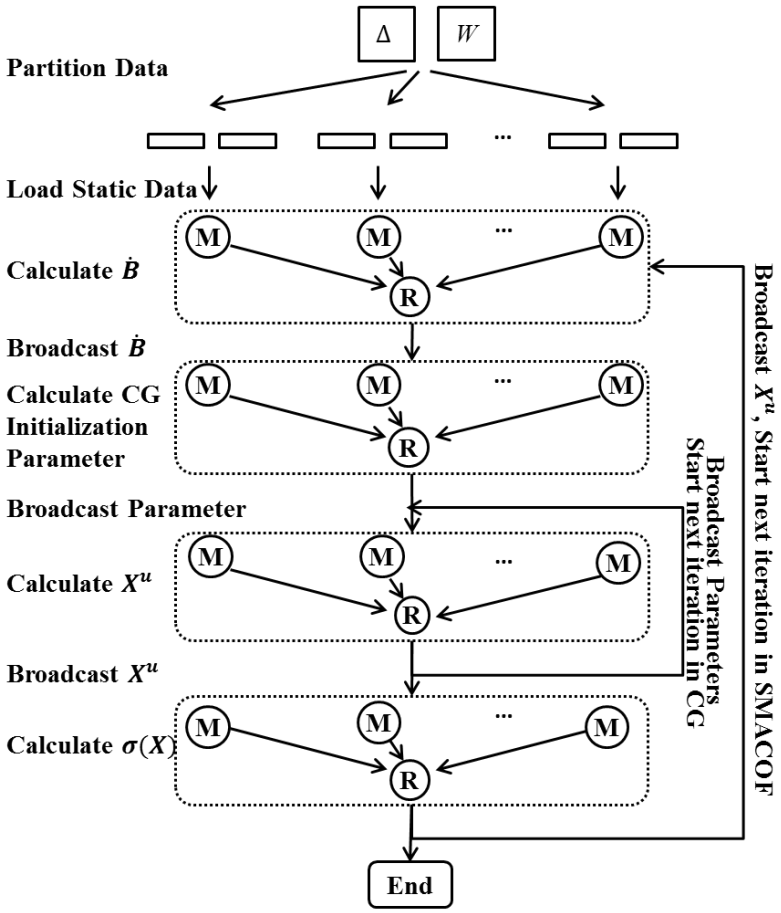

Figure 3.1: The flowchart of parallel WDA-SMACOF using an iterative MapReduce framework

3.3.3 **Fixed WDA-SMACOF**

The introduction of this scalable solution for weighted MDS problem has brought another possibility to solve a problem which is referred to as the unfolding problem in MDS. This problem is that if we have observed a part of data in the lower dimensional space and has their distances in the higher dimension, and the data points locations between each other wants to be fixed. One way of solving this is to use interpolation, as mentioned in Section 2. Interpolation considered the fixed data as *in-sample* data, and the varied data as *out-of-sample* data. Then by using the coordinates from the *in-sample* data, the out-of-sample points will be interpolated into the *in-sample* space one by one. This means the *out-of-sample* points are independent from each other, so no distances between *out-of-samples* is needed in this case. This is a descent solution if a lower time cost if more preferable and *in-sample* points already has all the structure included from *out-of-sample*

points in the low dimension space. But in some cases, when time is not essential and accuracy is more important, this solution more give bias answer. In our particular case, when a set of well-studied reference sequences are processed in order to generate coordinates in target dimension space, and some new sequences, which has a much larger size needs to be mapped into the same space, it is not practical to use interpolation. This is because these new sequences may has its own structure and sometimes new clusters that not shown in the reference sequences set may be found by clustering itself. And since interpolation does not keep the proximity in between the out-of-sample data itself, this information may be lost. And since the correlations between the reference sequence set and the new sequence set needs to be observed simultaneously, so the best way of doing that is to fix the reference sequence set and varies the new sequence set, so called fixed MDS. In order to generate robust result as well as keeping the scalability of fixed MDS, the WDA-SMACOF has to be updated as the following.

By expanding the object equation (1) with the temperature T, the function now is:

$$\sigma(X_T) = \sum_{i<j\leq N} w_{ij}\, \tilde{\delta}_{ij}^2 + \sum_{i<j\leq N} w_{ij}\, d_{ij}^2(X_T) - 2 \sum_{i<j\leq N} w_{ij}\, \tilde{\delta}_{ij} d_{ij}(X_T) \quad (25)$$

$$= \eta_{\tilde{\delta}}^2 + \eta^2(X_T) - 2\rho(X_T) \quad (26)$$

where the $\tilde{\delta}_{ij}$ is the distance in original dimension with termperature T and can be given in equation (10), and X_T is the N×L matrix where each row represents the coordinates under temperature T. For simplification of the equations, the X is used to denote X_T in following notations within this section. Note that part of coordinates in X is fixed in this algorithm, so denote the fixed part in X as X_1, and varied part of points in X as X_2, where their corresponding number is N_1 and N_2. So the X can be divided as shown in Figure 3.2. Note that $\eta_{\tilde{\delta}}^2$ is a constant, so only the second and third term of equation (26) needs to be modified.

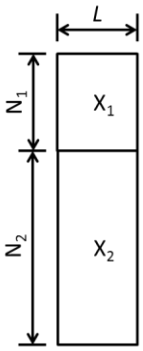

Figure 3.2: Graph representation of dividing X into X_1 and X_2

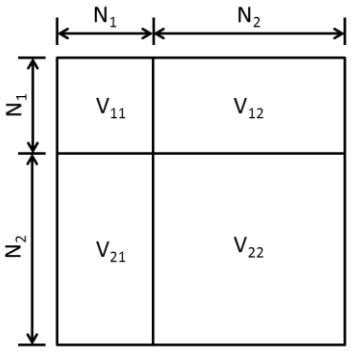

Figure 3.3: Graph representation of dividing V into 4 parts

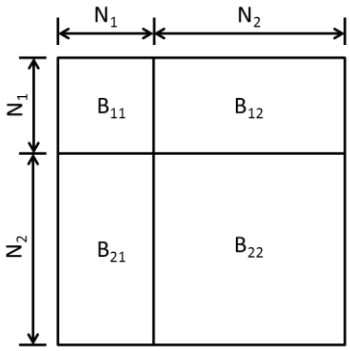

Figure 3.4: Graph representation of dividing B(Z) into 4 parts

Note that the second term means the square sum of distances between each pair of points in the target dimension under temperature T, so these distances are composed of 3 parts, the distances from points in X_1 to X_1, from points in X_2 to X_2, and from points in X_1 to X_2. And the second term $\eta^2(X)$ can be written as:

$$\eta^2(X) = trX'VX \qquad (27)$$

Where V is given in equation (13) and tr means the trace of a matrix. Consider that X is composed of X_1 and X_2, and V can be decomposed in the same way as in Figure 3.3, equation (27) can be then decomposed into

$$tr\ X'VX = trX_1'V_{11}X_1 + trX_2'V_{22}X_2 + trX_1'V_{12}X_2 + trX_2'V_{21}X_1 \qquad (28)$$

The matrices V_{11}, V_{22}, V_{12} and V_{21} are decomposed from the matrix V and shown in Figure 3.3. Note that all the diagonal values in V_{11} and V_{22} are as same as in V, so the first and second terms of equation (28) does not equal to $d(X_1)$ and $d(X_2)$. It also includes the values from distances between X_1 and X_2.

The third term of equation (26) is the sum of distances in between the mapping of X in the target dimension space multiplied by the distances in the original dimension space under temperature T. This term can be applied with *Cauchy-Schwarz inequality* [61], whereby an upper bound can be found as the following:

$$-2\rho(\mathrm{X}) = -2\mathrm{tr}\,\mathrm{X}'\mathrm{B}(\mathrm{X})\mathrm{X} \leq -2\mathrm{tr}\,\mathrm{XB}(\mathrm{Z})\mathrm{Z} \quad (29)$$

Where function B(X) and B(Z) can be calculated using equation (14). Note that Z is given as the previously estimated X, and in this case, X_1 remains fixed during iterations. So Z can be decomposed as X_1 and Z_2, where Z_2 means the previously estimated X_2. So in this equation, the part for X_1 actually equals to each other on the left and right of the equation. Therefore, denote B as for the matrix result from function B(Z), matrix B can be decomposed as same as in Figure 3.4. And left hand side of equation (29) is now:

$$tr\,X'B(X)X = trX_1'B_{11}X_1 + trX_2'B_{22}X_2 + trX_1'B_{12}X_2 + trX_2'B_{21}X_1 \quad (30)$$

Since the first term is fixed, only the rest of term in equation (30) needs to be updated with this inequality, which is now:

$$-2tr\,X'B(X)X \leq -2(trX_1'B_{11}X_1 + trX_2'B_{22}Z_2 + trX_1'B_{12}Z_2 + trZ_2'B_{21}X_1) \quad (31)$$

Therefore, by taking equation (28) and (31) into equation (26), the STRESS function can now be written as:

$$\sigma(X_T) = \eta_\delta^2 + trX_1'V_{11}X_1 + trX_2'V_{22}X_2 + trX_1'V_{12}X_2 + trX_2'V_{21}X_1 - 2(trX_1'B_{11}X_1 + trX_2'B_{22}X_2 + trX_1'B_{12}X_2 + trX_2'B_{21}X_1) \quad (32)$$

And the upper bound it is given as

$$\sigma(X_T) \leq \eta_\delta^2 + trX_1'V_{11}X_1 + trX_2'V_{22}X_2 + trX_1'V_{12}X_2 + trX_2'V_{21}X_1 - 2(trX_1'B_{11}X_1 + trX_2'B_{22}Z_2 + trX_1'B_{12}Z_2 + trZ_2'B_{21}X_1) = \tau(X_2, Z_2) \quad (33)$$

Since X_2 is the only varied part of this equation, by taking the partial derivative of equation (33), now it is:

$$\frac{\partial\tau(X_2,Z_2)}{\partial X_2} = (X_1'V_{12})' + V_{21}X_1 + 2V_{22}X_2 - 2B_{22}Z_2 - B_{21}X_1 - (X_1'B_{12})' \quad (34)$$

Since both V and B are symmetric matrices, so equation (34) now becomes

$$\frac{\partial\tau(X_2,Z_2)}{\partial X_2} = 2V_{21}X_1 + 2V_{22}X_2 - 2B_{22}Z_2 - 2B_{21}X_1 \quad (35)$$

By setting (35) to zero, the final majorization function is:

Algorithm 2 Fixed-WDA-SMACOF algorithm

Input: Δ, W, $\boldsymbol{X}_1$, X_2, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\alpha}$
Output: X as target dimension mapping
1: Generate random initial mapping X_2.
2: Compute $V_{21}X_1$
3: $\boldsymbol{k} = \boldsymbol{0}$;
4: while $\boldsymbol{T_k} \geq \boldsymbol{T_{min}}$ do
5: Compute $\boldsymbol{T_k}$ and $\tilde{\Delta}_{\boldsymbol{k}}$ using (12).
6: $\boldsymbol{t} = \boldsymbol{0}$;
7: while $\boldsymbol{\sigma}(\boldsymbol{X}^{|\boldsymbol{t}|}) - \boldsymbol{\sigma}(\boldsymbol{X}^{|\boldsymbol{t+1}|}) > \boldsymbol{\varepsilon}$
8: Use CG defined from (20) to (24) to solve (36).
9: $\boldsymbol{t} = \boldsymbol{t} + \boldsymbol{1}$;
10: end while
11: Cool down computational temperature $\boldsymbol{T_{k+1}} = \boldsymbol{\alpha}\, \boldsymbol{T_k}$;
12: $\boldsymbol{k} = \boldsymbol{k} + \boldsymbol{1}$
13: end while
14: $\boldsymbol{X}$ =output of SMACOF based on $\boldsymbol{T_k}$
15: return $\boldsymbol{X}$

$$V_{22}X_2 = B_{22}Z_2 - B_{21}X_1 - V_{21}X_1 \quad (36)$$

Recall the Z_2 is the estimated X_2 in previous iteration, this is now the final formula for fixed WDA-SMACOF. Note that $V_{21}X_1$ is fixed from the start of the algorithm, so it only needs to be computed once and stored in static memory. Instead of having order $N \times N \times L$ matrix multiplication in each iteration, now the calculation in each SMACOF iteration becomes $N_2 \times N_2 \times L$ and $N_2 \times N_1 \times L$. The CG calculation now is in order N_2 instead of N as well. Note that the V_{22} matrix is naturally a SPD matrix, so no approximation needs to be done beforehand for this algorithm. The detail of Fixed-WDA-SMACOF is given in algorithm 2.

3.3.4 **Parallelization of Fixed-WDA-SMACOF**

As the Fixed-WDA-SMACOF is an iterative application, so it is parallelized using iterative MapReduce. The parallelization of it is very similar to WDA-SMACOF, which involves nested iteration MapReduce job within one SMACOF iteration. The difference is that before the SMACOF iterations begins, matrix multiplication of $V_{21}X_1$ needs to be done. And in each SMACOF iteration, two matrix multiplications are done instead of just one before the CG computation begins. The added matrix multiplication is for the calculation of $B_{21}X_1$. Note that if N_1 is relatively small, this calculation could be finished within the main driver using only one compute node to avoid communication overhead. But for most cases, this computation is done in

parallel. The $\dot{B}_2$ represents the right hand side of equation (36). And all the other operations in this computation can be finished within main driver since they are linear operations.

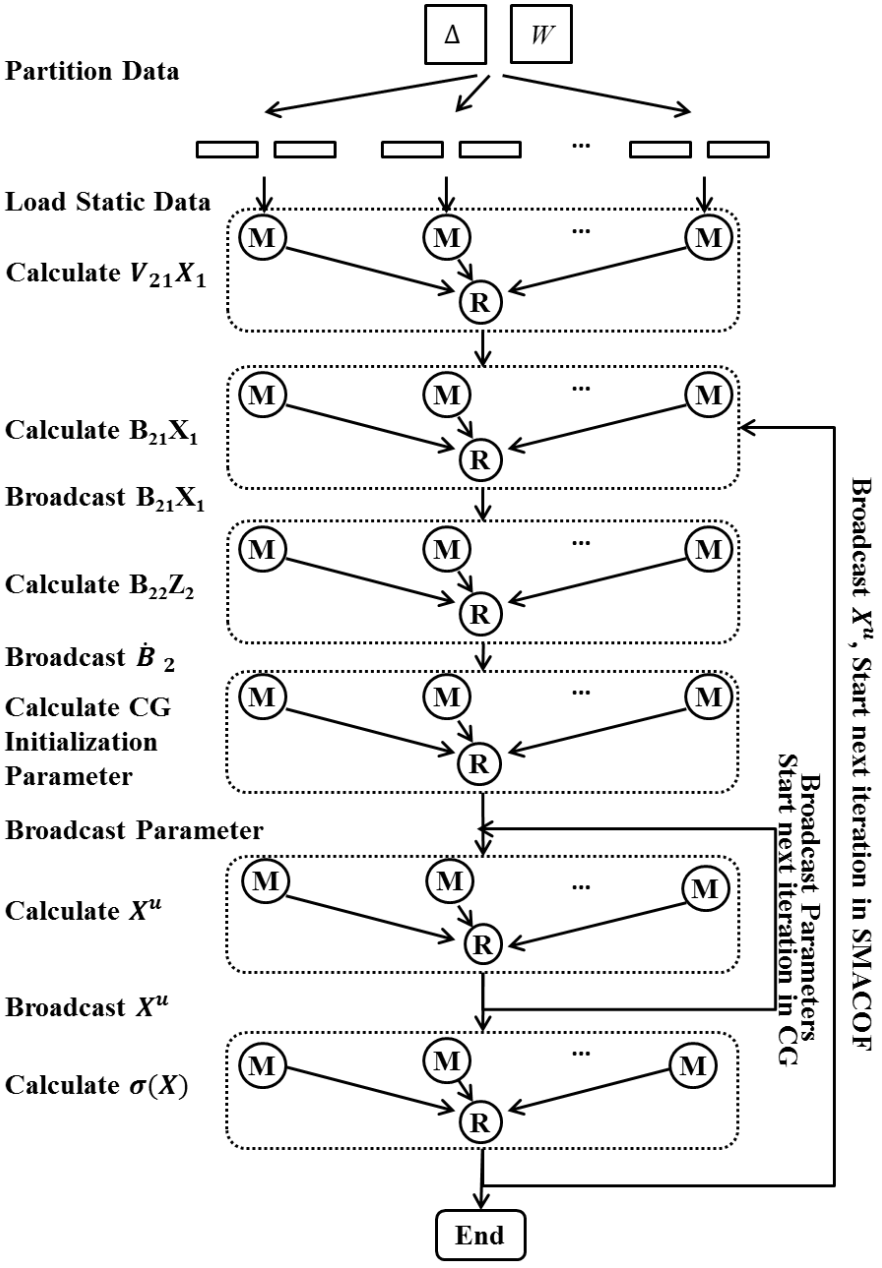

Figure 3.5: The flowchart of parallel Fixed-WDA-SMACOF using an iterative MapReduce framework

3.4 **Performance Analysis**

The experiments were carried out on two cluster environment. One is the FutureGrid XRay Cluster. It is a Cray XT5m super computer and can use cluster compatibility mode to simulate the environment on multi-node computer cluster, where each node has 8 cores and 16GB memory. In total, XRay has 168 AMD Opteron 2378 CPUs and 1324 cores. The network system of XRay uses Cray seastar interconnect. The other cluster is the Indiana University BigRed II cluster. Big Red II is Indiana University's main system for high-performance parallel computing and it is one of the world's fastest research supercomputers. It is a Cray XE6/XK7 supercomputer with a hybrid architecture providing a total of 1,020 compute nodes. All of these compute nodes are connected through the Cray Gemini interconnect. One can also use cluster compatibility mode to reserve up to 128 nodes. These CPU-only nodes each contain two AMD Opteron 16-core CPUs and 64 GB of memory. There are also GPU nodes available on BigRed II but not used for the performance analysis.

The dataset used here includes the artificial hmp16S rRNA (Artificial RNA), real hmp 16S rRNA, Clusters of Orthologous Groups of proteins (COG Protein) and arbuscular mycorrhizal fungus (AM Fungi DNA) data. The artificial RNA is used to test the clustering and visualization result from DACIDR, and compared it with traditional clustering methods. So this dataset has a relatively smaller size of sequences compared to other dataset, and the data points are well clustered. Real hmp16S rRNA [46] dataset is the real large dataset that needs to be clustered and visualized. The OTUs within the dataset are unknown and needs to be found. COG Protein [62] dataset is used from the project lead by the National Center for Biotechnology. The dataset is well classified by experts manually, and DACIDR was applied on it to verify the results. AM Fungi [63] DNA data are sequences collected using two different 454 technology sequencing. The goal is to cluster this sequences then find the representative sequences from each cluster. Finally to use phylogenetic analysis to identify the relationships of these sequences with some well classified fungal sequences. Each dataset includes different number of sequences listed in Table 2:

Table 2 The dataset used in the experiments across the dissertation

	Artificial RNA	**hmp16S rRNA**	**COG Protein**	**Fungi DNA**
Total Number of Sequences	100k	1.1 million	183k	957k
Number of Unique Sequences	4640	684769	183k	446k

Note that each dataset has duplicates sequences in the original dataset. So we always clean the dataset by removing the duplicates and only keeping the unique sequences inside the dataset. When the size of the dataset is stated further in this dissertation, it will always be referred to the number of unique sequences in that dataset.

In the experiments for this performance analysis, four different SMACOF algorithms were compared as shown in Table 3. All of these dissimilarities used as input for the experiments were generated using ASA and SWG alignment. The DA means the algorithm is optimized using DA techniques, where W stands for weighted. So WDA-SMACOF is the algorithm proposed and NDA-SMACOF means the non-weighted DA-SMACOF algorithm described in Section 2.3. The WEM-SMACOF is the SMACOF with the matrix inversion and without DA optimization as proposed in [11]. The NEM-SMACOF is the original SMACOF function which assumes all weights equal 1 in all circumstances. Additionally, equation (10) shows that all the EM cases could be considered as a special case of DA algorithms that initial temperatures were set to 0.

Table 3 All 4 algorithms tested in following experiments

	DA	**EM**
Weight	WDA-SMACOF	WEM-SMACOF
Non-Weight	NDA-SMACOF	NEM-SMACOF

The performance analysis includes 4 sections: the first 3 sections were based on all varied WDA-SMACOF and the last section is about fixed WDA-SMACOF. In detail, first section is about accuracy comparison between WDA-SMACOF and other there popular MDS methods. The second section is the time cost analysis on WDA-SMACOF, with comparison from other MDS methods as well. The third section is the parallel efficiency analysis on WDA-SMACOF itself

with iterative MapReduce framework. The fourth section is the accuracy analysis of fixed WDA-SMACOF compared with interpolation.

3.4.1 **Accuracy Comparison of WDA-SMACOF**

We tested the accuracy of the results based on normalized STRESS value, which can be calculated by

$$\overline{\sigma}(X) = \sum_{i<j\leq N} w_{ij} \frac{(d_{ij}(X)-\delta_{ij})^2}{\delta_{ij}^2} \quad (37)$$

where δ is given by PID distance calculated from pairwise sequence alignment. Equation (37) is least squares sum of difference between the mapped distance after dimension reduction and original distance and naturally lower normalized STRESS means better performance [11].

All of these tests were done with 20 runs for each algorithm. The threshold ε is set to 10^{-6}. The results were based on the average of these runs. The error bars in the figures were the maximum and minimum value in the runs. In general, if the average of the normalized STRESS is lower, the accuracy is higher. And if the error bars are nearer to the average of an algorithm, the robustness of the algorithm is higher. The target dimension L is set to 3, so that the dimension reduction result can be visualized.

The result from Metagenomcs dataset includes 2000 of unique sequences. As this dataset has a small number of sequences, the sequential versions of these 4 algorithms were tested. This sequences were aligned using local alignment algorithm to calculate the original dissimilarity. And during that calculation, 10.775% of the original distances values were found as missing because of the low alignment quality. From the result shown in Figure 3.6, we observed that both of the weighted solutions outperforms the non-weighted solution. DA solutions showed much less divergence compared to EM solutions. The average normalized STRESS value for Full MDS was 0.0439, which outperforms non-weighted cases by 23%. The visualization result from WDA-SMACOF is shown in Figure 3.7, and it is clear that in this dataset, all clusters are clear separated.

Differently from DNA and RNA data mentioned above, the Protein data doesn't have nicely clustered structure after dimension reduction as shown in Figure 3.9, and its distance calculation

was based on global alignment other than local alignment. Before the MDS algorithms started, a distance transformation is done on the dissimilarities in order to generate result with more clustering information. In our experiments, we used 4872 consensus sequences to run full MDS, and interpolated rest 95672 sequences to these consensus sequences. Among these distances from Full MDS and Interpolation, 10% of them were randomly chosen to be missing distances. The runs for 4872 in-sample sequences were carried out on a single core, while the Interpolation for 95672 out-of-sample sequences used 40 cores. The results for COG Protein data were shown in Figure 3.8. Non-weighted and weighted cases show insignificant difference that WDA performs only 7.2% better than non-weighted cases.

The original hmp16SrRNA dataset has 680k unique sequences. In these experiments, we selected 10k of it for this experiment. Due to the larger size, it cannot be done on a single core, so we used the parallel version of Full MDS and Interpolation to run the experiments on 80 cores. The distance was calculated using local alignments and 9.98% of distances were randomly missing and set to an arbitrary number. The normalized STRESS was shown in Figure 3.10. In this case, the weighted solutions have a normalized STRESS value lower than non-weighted solutions by 40%. The visualization result of WDA-SMACOF shows that the clusters of hmp16SrRNA are not as clearly separated as in the Artificial RNA data, but the clusters are still able to be separated by naked eyes.

These accuracy analysis were based on three different dataset varies from DNA, RNA to Protein. And all of them show consistent result that WDA-SMACOF always has the lowest STRESS value, i.e. highest accuracy compare to other existing methods. These three dataset size varies from 2k to 10k, and a relatively large data test were carried out on 600 cores on FutureGrid xRay to compare the normalized STRESS value for 100k data selected from AM Fungal data. As the same in previous experiments, 10% of the distances from original distance matrix were missing. This result also shows that the WDA-SMACOF has the lowest STRESS value. The average STRESS value for WDA-SMACOF is 0.0153, which is lower than NDA-SMACOF by 0.022, and notice the NDA-SMACOF is also robust without much variance on different runs. But is still has a much higher STRESS compared to the weighted solutions. NEM-SMACOF has the worst

performance as always, and WEM-SMACOF is less robust than the WDA-SMACOF solution, and has a STRESS value 7.9% higher than the result from WDA-SMACOF.

In these experiments, different dataset shows different features after dimension reduction. These 4 algorithms has been tested using 4 different dataset, and with different size. Although these three dataset had diverted visualization results, WDA-SMACOF always shows lowest normalized STRESS value and smallest divergence in all experiments. It is safe to say that WDA-SMACOF is more accurate and robust than the previously proposed SMACOF and DA-SMACOF methods with a non-trivial weight function support.

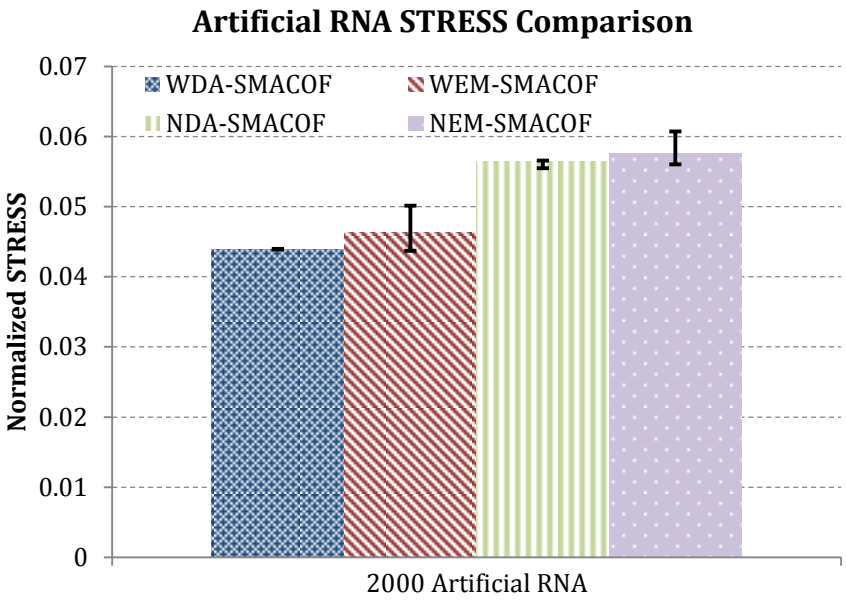

Figure 3.6: The normalized STRESS value comparison between 4 MDS algorithms using 2000 Artificial RNA sequences. All 4 algorithms in this experiment are sequential.

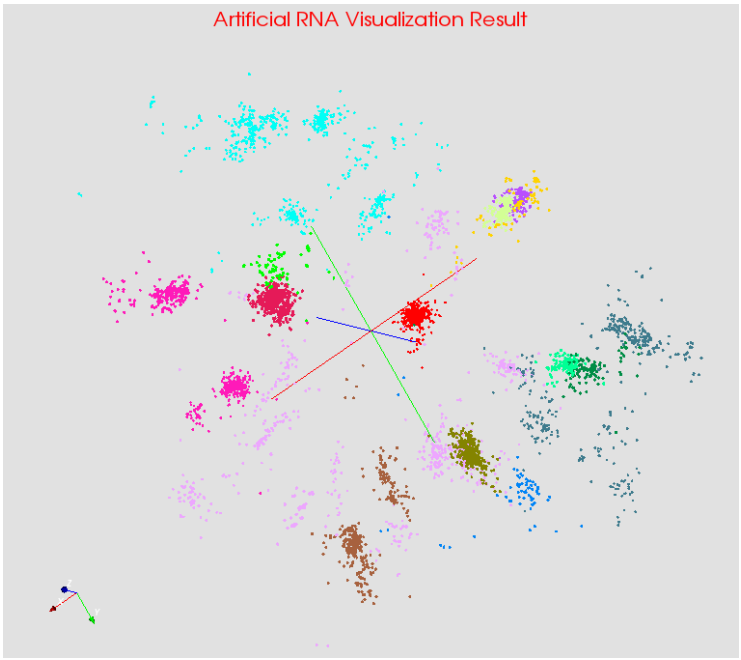

Figure 3.7: The clustering and visualization result of entire Artificial RNA dataset with 13 clusters labeled. Each points belongs to the same color is a cluster found by using DA-PWC program.

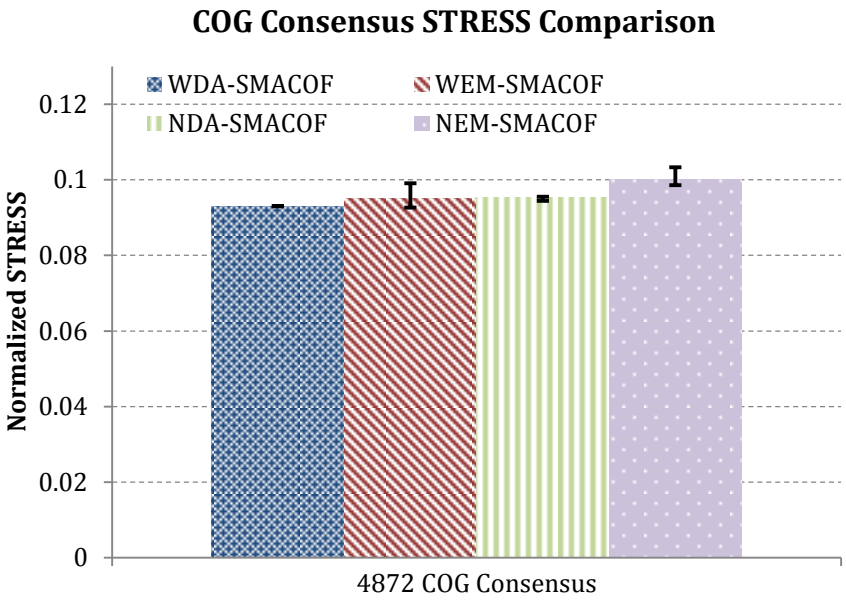

Figure 3.8: The normalized STRESS value comparison between 4 MDS algorithms using 4872 COG consensus sequences. All 4 algorithms in this experiment are sequential.

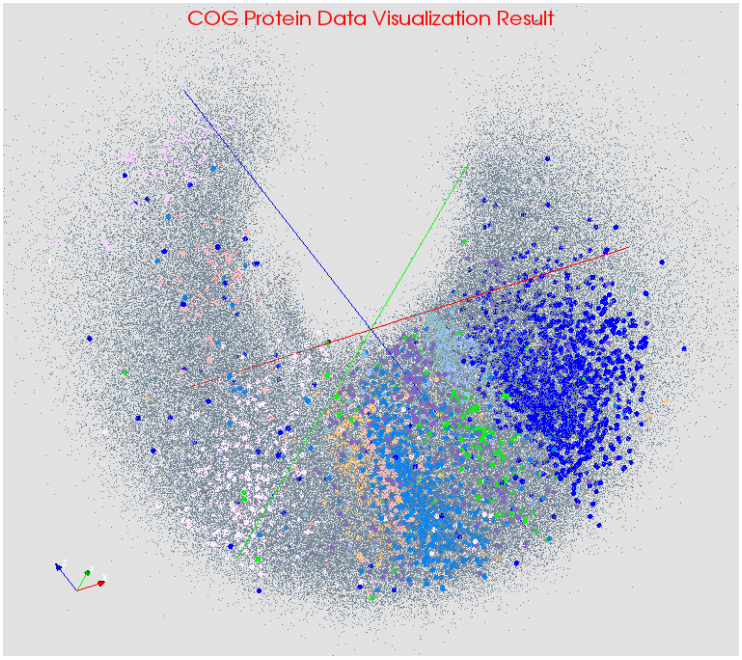

Figure 3.9: The clustering and visualization result of entire COG dataset with a few clusters labeled. These clusters were manually defined by using the information from NIH.

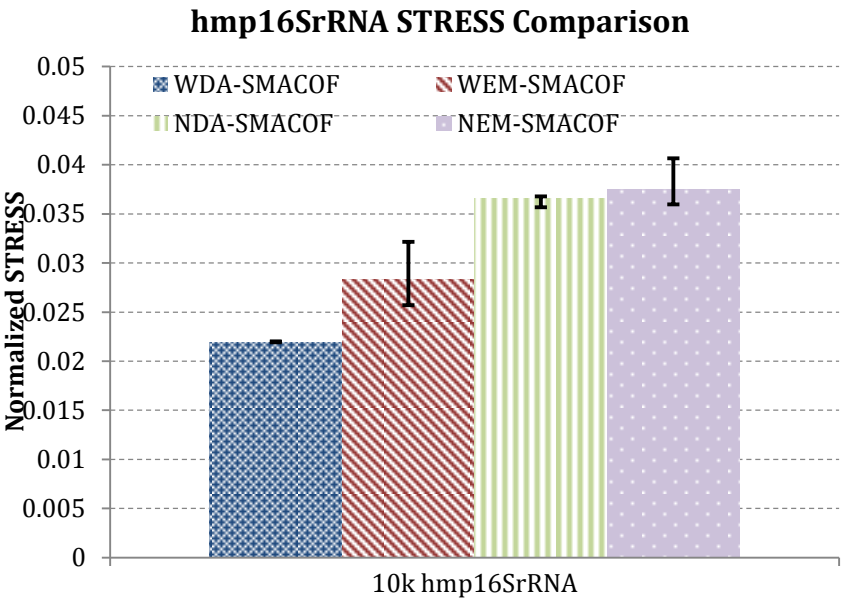

Figure 3.10: The normalized STRESS value comparison between 4 MDS algorithms using 10k hmp 16S rRNA sequences. All 4 algorithms in this experiment are parallelized using Twister on 80 cores.

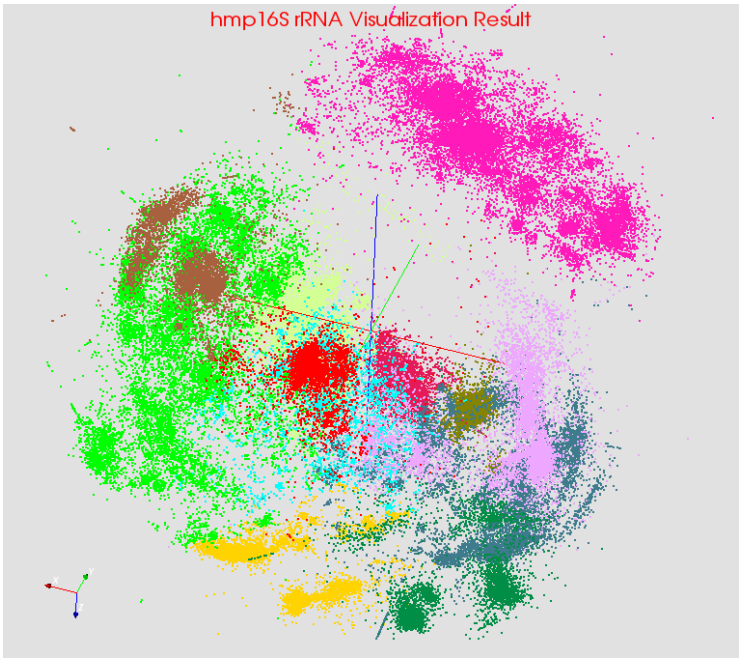

Figure 3.11: The clustering and visualization result of entire hmp16S rRNA dataset with 11 mega regions labeled. Each points belongs to the same color is a mega region found by using DA-PWC program.

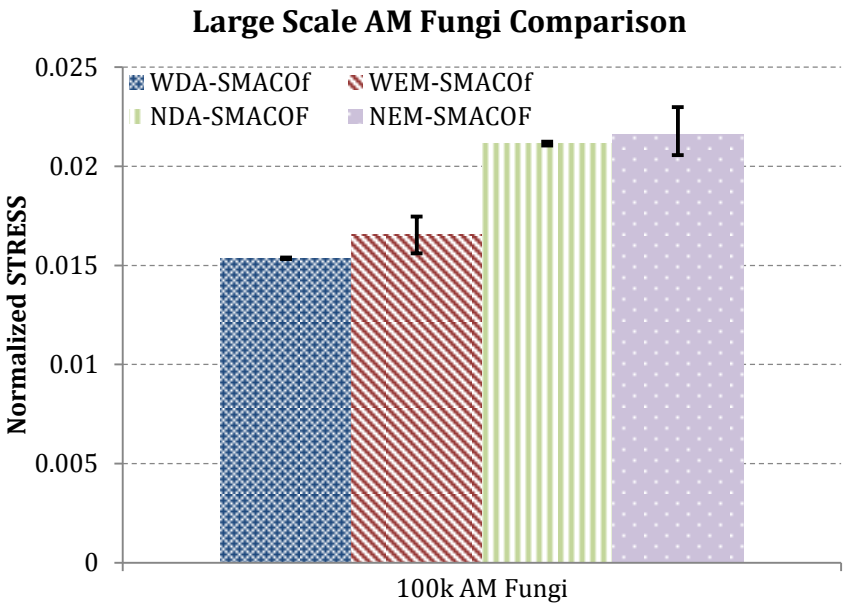

Figure 3.12: The normalized STRESS value comparison between 4 MDS algorithms using 100k AM Fungal sequences. All 4 algorithms in this experiment are parallelized using Twister on 600 cores.

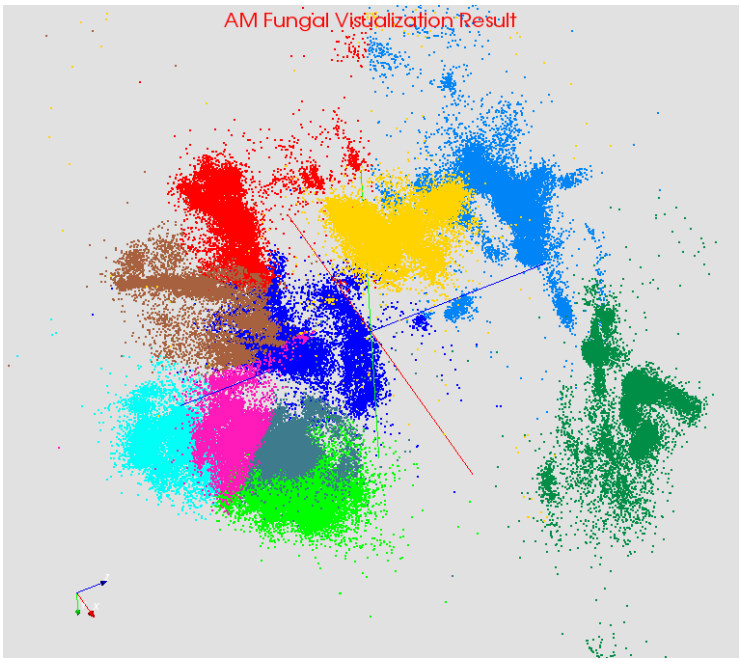

Figure 3.13: The clustering and visualization result of entire AM fungal dataset with 10 mega regions labeled. Each points belongs to the same color is a mega region found by using DA-PWC program.

3.4.2 **Time Cost Analysis of WDA-SMACOF**

The time cost of WDA-SMACOF can be divided into two parts: first is the normal SMACOF computation happens within one SMACOF iteration, the other part is the CG computation. Note that all three other algorithms, NDA-, WEM- and NEM-SMACOF has the same computations as in WDA-SMACOF, but without CG computation. And as there are multiple CG iterations per SMACOF iteration in WDA-SMACOF, the computation time will be dominate by the number of CG iterations per SMACOF iteration. So the time cost comparison are divided into two parts for WDA-SMACOF: first is to compare the equivalent computations of all 4 algorithms within SMACOF iteration; then do the analysis on the nested loop for CG computations for WDA-SMACOF only.

SMACOF computation: The time cost of SMACOF involves the computation of function B(Z), the matrix multiplication of B(Z)×Z and the STRESS calculation. The time cost of 2000 Artificial RNA, 4872 COG consensus and 10k hmp16SrRNA are compared and analyzed here. For the experiments result shown here, the ending condition were using threshold, so that the iteration number could be various due to the configuration/feature space of different dataset.

2000 sequences selected from Artificial RNA dataset were used for experiments over a single core. Figure 3.14 shows that WDA-SMACOF and WEM-SMACOF has a higher time cost compare to WEM-SMACOF and NEM-SMACOF. This is because by using DA optimization, the real number of iterations for these two algorithms is larger than those two latter algorithms. And WDA-SMACOF is two times slower than WEM-SMACOF algorithm and 37% faster than the NDA-SMACOF.

The parallelized result in 10k hmp16SrRNA data shows the similar result in Figure 3.15. All these 4 algorithms show similar behaviors with the sequential version. WDA-SMACOF is still the most robust result. And it is 68% faster than the NDA-SMACOF version. Different from the result on 2000 Artificial RNA data, the WDA-SMACOF if faster than NEM-SMACOF on this 10k result., and it is also 2 times faster than WEM-SMACOF.

The test on the 4872 COG consensus sequences shows an interesting and different result trend from all previous result as shown in Figure 3.16. Although WDA-SMACOF is still the most robust result, where the time cost variance of it is very small, it is the slowest algorithm among all 4 algorithms. So to investigate more into the result, the normalized STRESS value for this run on every 40 iteration is shown in Figure 3.17.

In this result, it shows that the normalized STRESS values in both WDA-SMACOF and NDA-SMACOF are reduced fast during the first 200 iterations. The NDA-SMACOF (T), which is the "True" normalized STRESS value seen by NDA-SMACOF during the iterations considering all weights equal 1. Since the threshold is set as the same for both of the algorithms, so the NDA-SMACOF may converge faster if all distances are considered, as in this case, the STRESS value for NDA-SMACOF (T) is 0.120240360398106 at iteration 440 and 0.120242355, the difference for them is 8.00509E-06, which is smaller than the threshold set as 10E-5, so it will converge at iteration 480. However, for the WDA-SMACOF, the STRESS value at iteration 440 is 0.093093143 and at iteration 480 is 0.093048041, and their difference is 4.51021E-05 and it is much larger than the threshold. So it won't be able to converge for more iterations. However, the STRESS value for NDA-SMACOF calculated with the correct weight matrix (where missing distances has a weight 0), is going up and down around 0.095, and cannot converge with more iterations from iteration 320. This is because the NDA-SMACOF derives from the STRESS function with all weights equal 1, so it cannot converge with a weight matrix that has zeroes inside. So although sometimes NDA-SMACOF (T) converges faster, their result is approximated result but not as accurate as WDA-SMACOF. And the final STRESS value for them is not from converged result but rather than approximation.

After studying the time cost for fixed threshold runs, the next question would be how about the result if we fix the iteration numbers. The result from fixed iterations runs is shown in Figure 3.18.

This test is done using 10k hmp16SrRNA data on 80 cores from xRay at Futuregrid. The percentage of missing distances increases during the runs. The WDA-SMACOF and NDA-SMACOF use the same configuration during each run, and the since the iteration number are the

same, this is actually the head to head comparison for only the computation from STRESS calculation, and the majorization function. The NDA-SMACOF always has the same time cost around 206 seconds, and WDA-SMACOF has a stable time cost lowered when percentage of missing distances increases. This is because according to equation (14), if a weight correspond to a distance is set to zero, the computation of B(Z) and STRESS value on that function could be saved. Also, if the most of the distances were zero, the matrix multiplication is similar to the sparse multiply, which can reduce a majority of time cost. And for the case the 90% of the distances are missing, the time cost is reduced by 42% at most. The rest time cost mainly came from the computation in the main driver and the communication overhead.

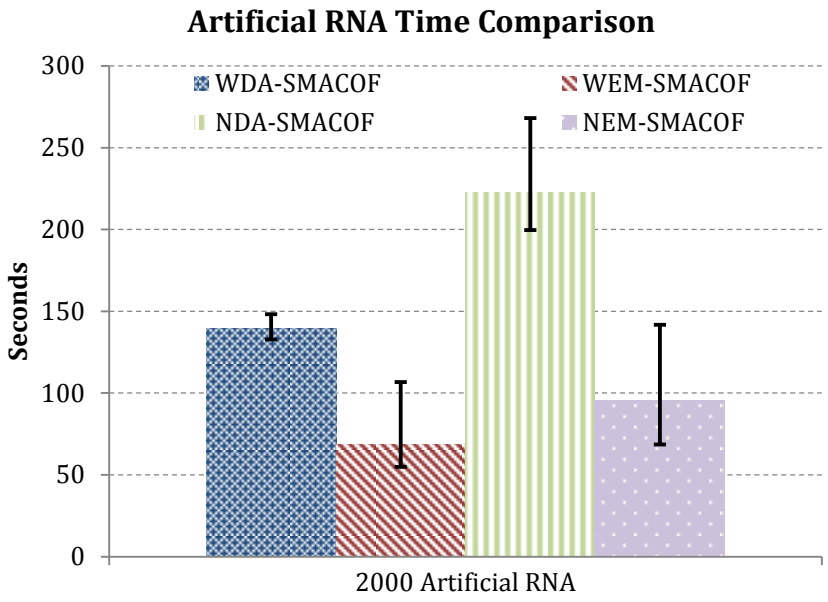

Figure 3.14: The time cost comparison between 4 MDS algorithms using 2000 Artificial RNA sequences. All 4 algorithms in this experiment are sequential.

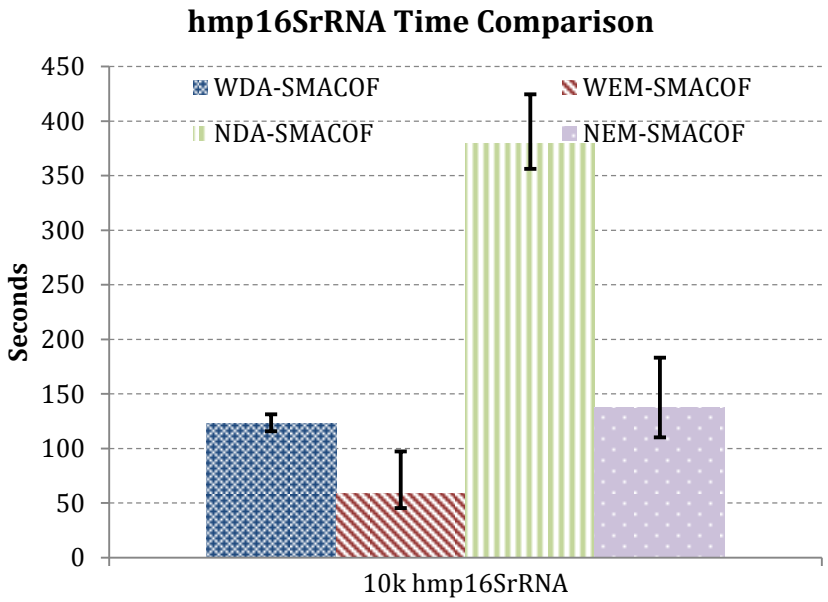

Figure 3.15: The time cost comparison between 4 MDS algorithms using 10k hmp 16S rRNA sequences. All 4 algorithms in this experiment are parallelized using Twister on 80 cores.

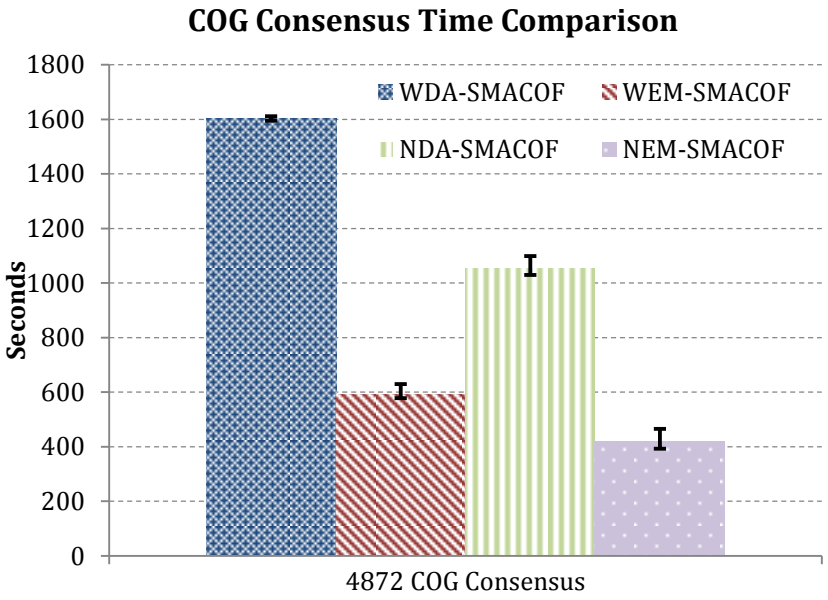

Figure 3.16: The time cost comparison between 4 MDS algorithms using 4872 COG consensus sequences. All 4 algorithms in this experiment are sequential.

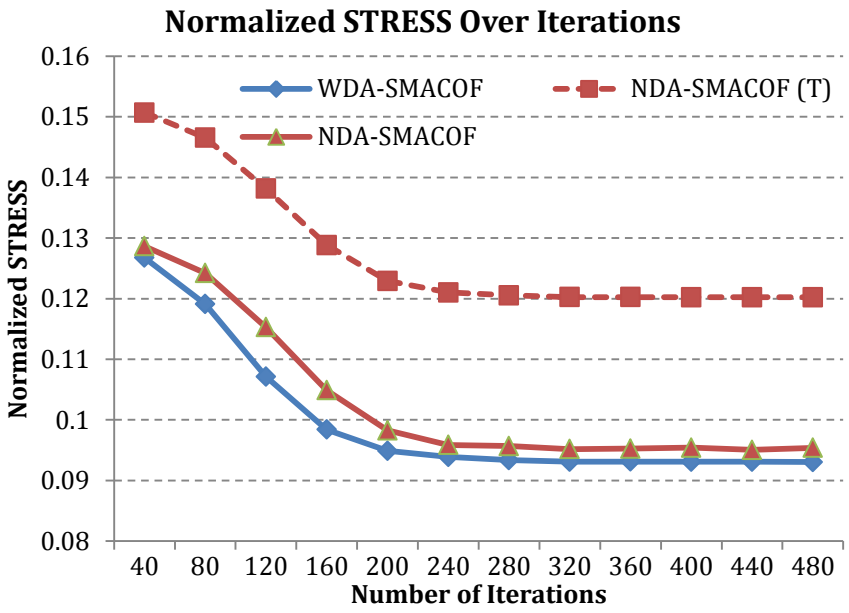

Figure 3.17: The normalized STRESS with increasing number of iterations between a weighted MDS algorithm WDA-SMACOF and a non-weighted MDS algorithm, NDA-SMACOF. The NDA-SMACOF (T) is the actual normalized STRESS value that calculated using the weight matrix, where NDA-SMACOF is the STRESS value with weights all equal 1.

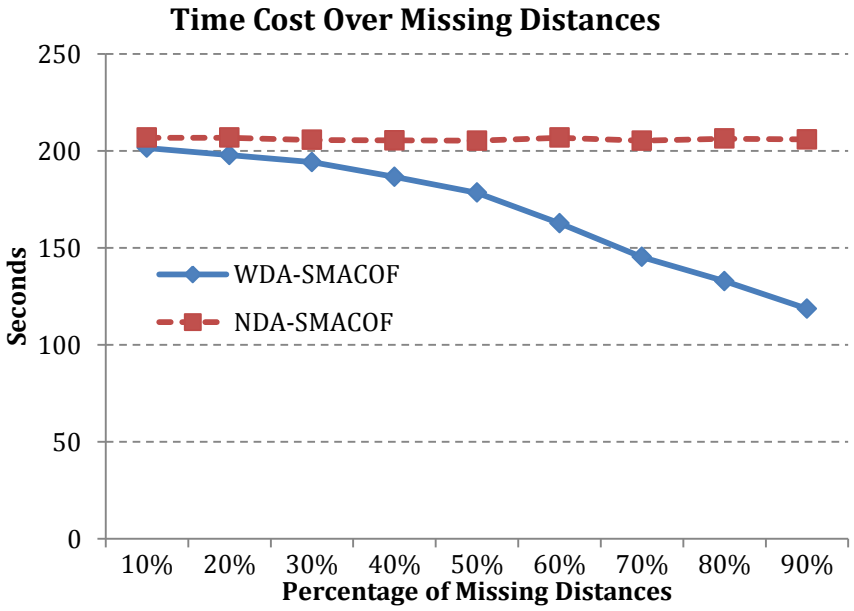

Figure 3.18: The time cost of WDA-SMACOF and NDA-SMACOF processing 10k hmp 16S rRNA data using 80 cores by fixing the iteration to 400 and increases the percentage of missing distances randomly.

CG Computation: The CG computation for WDA-SMACOF is essential as this is the main contribution for this algorithm that enables quadratic time complexity. In this set of experiments, the CG computation versus matrix inversion is tested, and the number of CG iterations needed per SMACOF iteration is tested. The time cost detail of WDA-SMACOF is also studied.

As shown in Figure 3.19, the time cost of matrix inversion is lower than CG computation when the data size is smaller than 4k. So it is obvious that the original SMACOF with matrix inversion has a lower time cost than WDA-SMACOF. The CG requires averagely 20 iterations to solve the Ax=b formula on the data selected here. An error is calculated with a threshold of 10^{-6} to the norm of the residual calculated in each CG iteration as suggested by [64]. As the data size increases, the matrix inverse performs much slower than CG, this is because matrix inversion has $O(N^3)$ time complexity and CG only has $O(N^2)$ time complexity. So when the data size increases, if the number of CG iterations does not increases as same as the matrix inversion, its time cost will be much lower. As for 4k data, CG and matrix inversion has the same time cost, but when the data size increases to 8k, CG perform 2.5 times faster than matrix inversion. And it could be expected that when data size increases to even larger size, the differences in the time cost with these two computations will become larger.

The number of CG iterations needed per SMACOF iteration in WDA-SMACOF is a key factor that dominates the total computation time for WDA-SMACOF. If the weights all equals 1, by increasing the data size alone, the CG iterations needed per SMACOF iteration is around 2 despite of the increasing data size. If the data size is fixed, and randomly chosen part of the distances as missing values, the number of CG iterations needed per SMACOF iteration is shown in Figure 3.20. It shows that if the weights are uniformly selected as missing, the CG iterations per SMACOF iteration won't increases that rapidly. Even when the percentage of missing distances increases to 50%, the average CG iteration per SMACOF iteration is around 4, which is much less than the total size of data as 100k tested here. Figure 3.21 shows that by taking the average of all situations with percentage of missing distances increasing from 0.1 to 0.5, the number of CG iterations only increases from 3.35 to 3.74 with the data size increases from 20k to 100k. This is because the convergence rate of CG really depends on the eigenvalues found for the matrix on the

left hand side of the formula Ax=b. If A has an evenly distributed eigenvalue, which means the direction found in the first CG iteration will make the cost function reaches a point where it is near the final destination. However, if the eigenvalues for A has property that the largest eigenvalue is much larger than the smallest eigenvalue, the number of iterations will be much larger. And during the study of this dissertation, this condition is very unlikely to happen by testing several of scenarios occurred with the dissimilarities generated using sequence alignment.

Sammon's Mapping is a popular non-linear MDS algorithm that solves equation (2) instead of equation (1). WDA-SMACOF can also solve this type of problem by leveraging the power of weight function. As shown in equation (2), SSTRESS can be converting to equation (1) with an updated weight, and the weight can be given as $w_{ij}^{u} = w_{ij}/\delta_{ij}$. The weight matrix can then be calculated beforehand, and the eigenvectors of the given matrix V will be a lot different from the weight with only zeroes and ones. The time cost of WDA-SMACOF solving Sammon's STRESS will be higher than solving a randomly generated weight matrix because of this. And the number of CG iterations per SMACOF iteration is given by taking the average of total number of CG iterations divided by total number of SMACOF iteration. The result shows that despite of various dataset, the CG iterations needed per SMACOF iteration are around 200. Figure 3.22 illustrates the average number of CG iterations per SMACOF iteration needed for Sammon's Mapping, and it shows that for COG Consensus data the number of iterations needed are the smallest, which is around 110. The number of iterations for hmp16SrRNA data is the most among all three dataset, which is around 200. And for 2000 Artificial RNA data, the number of iterations is also around 170, which is only 11.3% less than processing the hmp16SrRNA data. But the data size of Artificial RNA data is only 1/5 of the 10k hmp16SrRNA data. This is because the number of CG iterations are mainly dominated by the eigenvectors of matrix V, rather than the data size. The following runs were finished using 600 cores on Futuregrid xRay. Figure 3.23 shows the average number of CG iterations per SMACOF iteration for WDA-SMACOF when using two different weightings. When use the weights all equals 1, the number of CG iterations per SMACOF iteration is always around 2 or 3. And for Sammon's Mapping, the number of CG iterations increases to around 200. However, even by increasing the data size 5 times with a same dataset,

the number of CG iterations needed per SMACOF iteration only increases 1.4 times. One can assume that by increasing the data size won't increase computation time much because of this even with larger dataset as long as the dataset has a similar visualization data structure with Sammon's Mapping. Figure 3.24 shows the time cost of WDA-SMACOF with two different weighting processing the same dataset, and the number of SMACOF iterations were set to 100. And the time of WDA-SMACOF increases because of the number of CG iterations. It shows that WDA-SMACOF takes around 21 more time to finish Sammon's Mapping than dealing with the case where all weights equal 1 on 20k data. And it takes around 27 times longer when data size increases to 40k and 60k. Finally, in 80k and 100k cases, it takes 32 times longer to finish. So when processing with Sammon's mapping, WDA-SMACOF takes around 20 to 30 times longer to finish than dealing with trivial weights matrices.

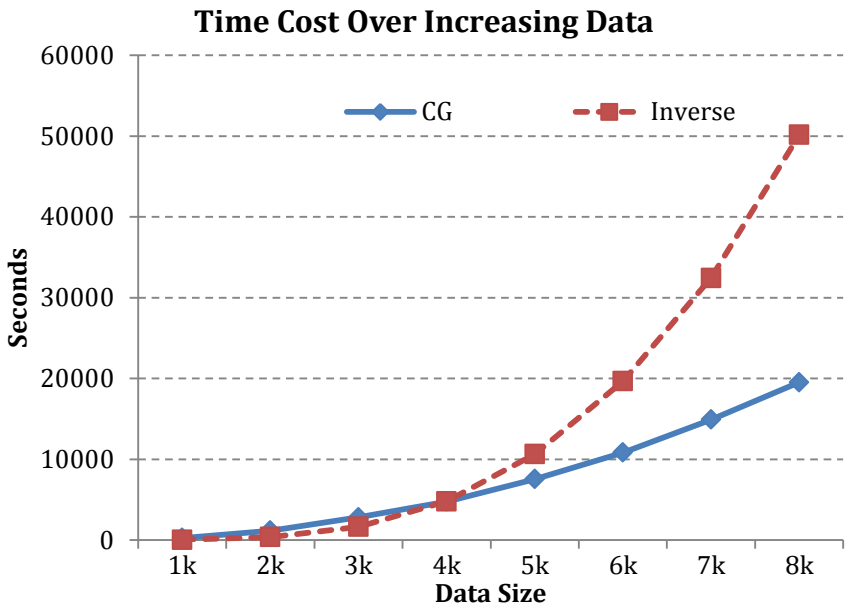

Figure 3.19: The time cost of CG versus matrix inverse over 1k to 8k hmp 16S rRNA data. The matrix inversion uses Cholesky Decomposition and CG uses 20 iterations. Both algorithms were sequential.

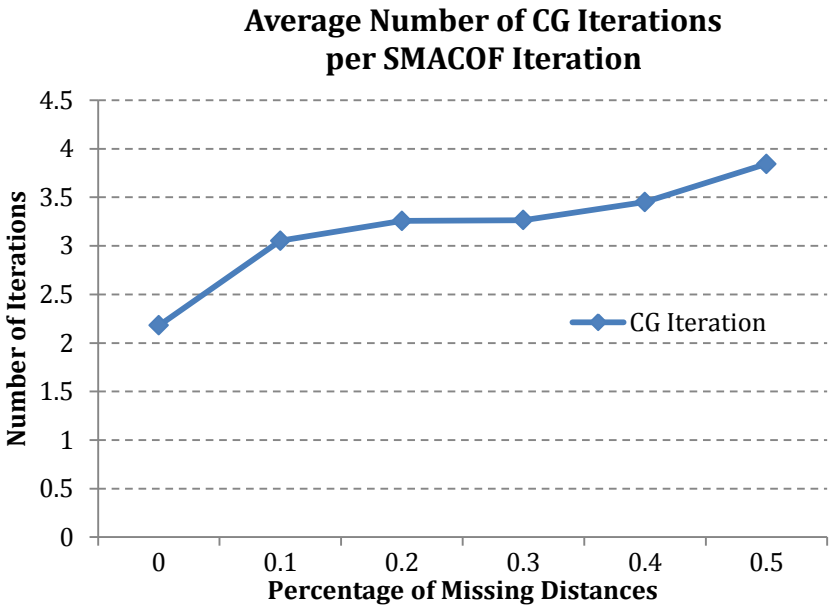

Figure 3.20: The number of CG iteration needed for 100k AM Fungal Data processed with parallel WDA-SMACOF. The percentage of missing distances increases from 0 to 0.5 and all missing distances are randomly chosen.

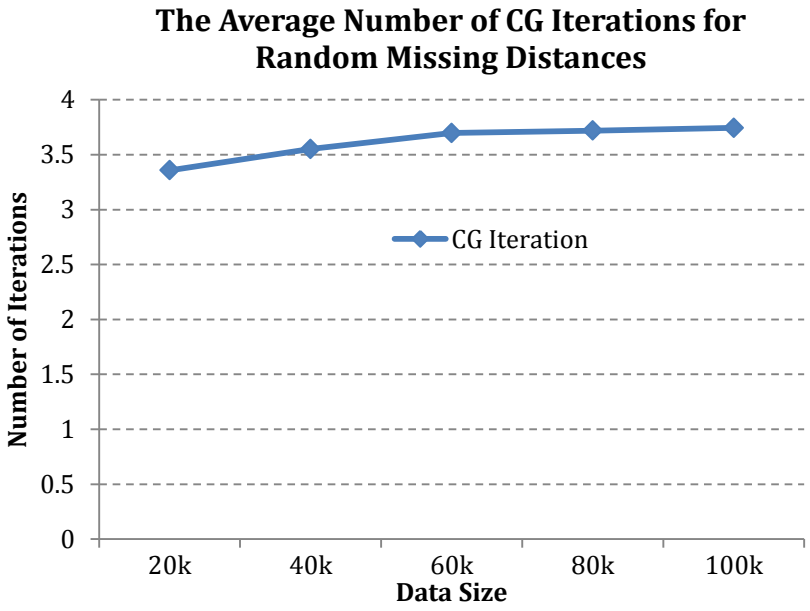

Figure 3.21: The number of CG iteration needed for AM Fungal Data processed with parallel WDA-SMACOF. The data size varies from 20k to 100k. The number of iterations took over the average of number of CG iterations from the scenarios that percentage of missing distances increases from 0 to 0.5.

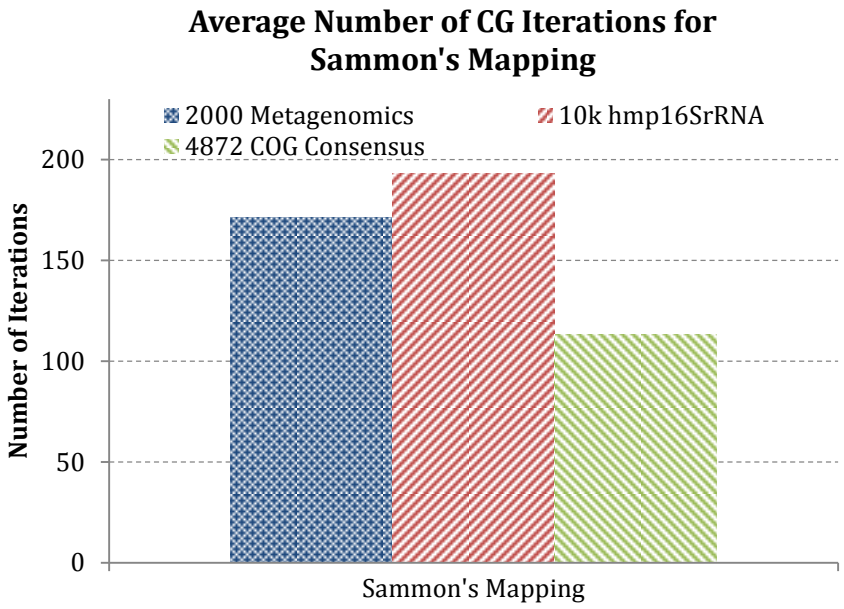

Figure 3.22: The number of CG iteration needed for 2k Artificial RNA data, 4872 CGO Protein data and 10k hmp16S rRNA data processed with parallel WDA-SMACOF with Sammon's Mapping.

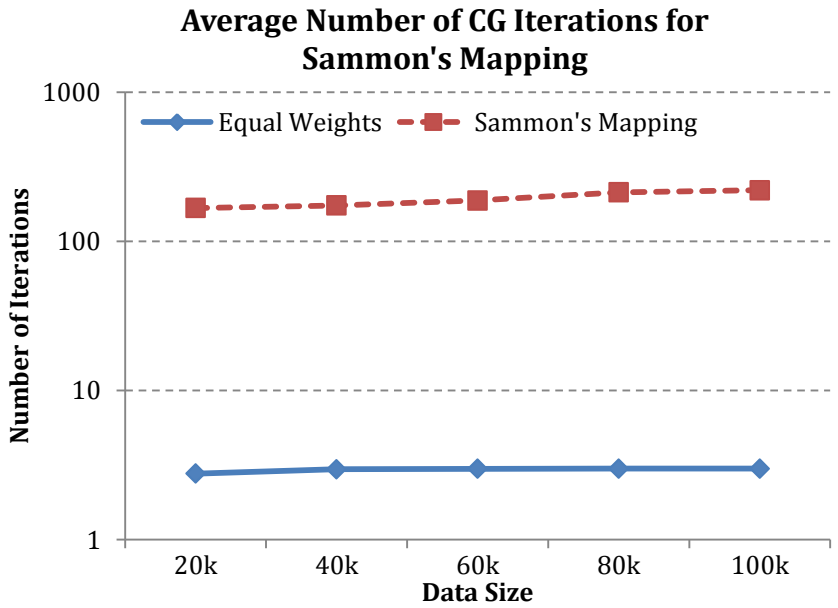

Figure 3.23: The number of CG iteration needed for AM Fungal data processed with parallel WDA-SMACOF with Sammon's Mapping. The data size increases from 20k to 100k.

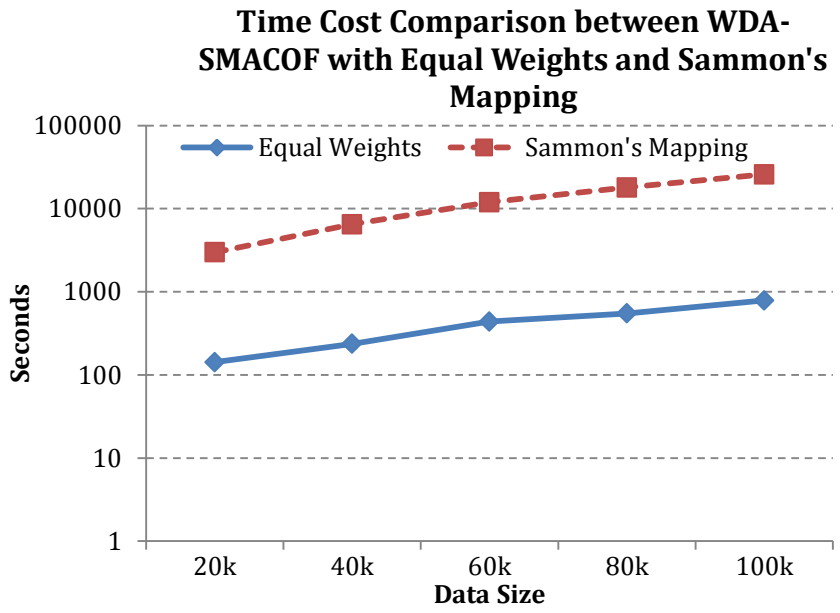

Figure 3.24: Time Cost of WDA-SMACOF with equal weights compared with WDA-SMACOF with Sammon's Mapping using AM Fungal data. The data size increases from 20k to 100k. Each run takes 100 SMACOF iterations.

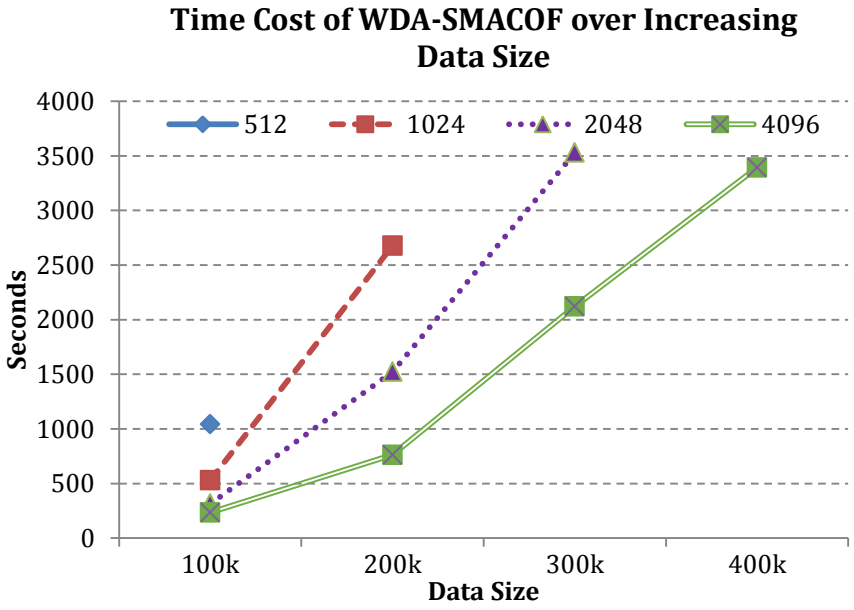

Figure 3.25: The time cost of parallel WDA-SMACOF by fixing the number of processors and increase the data size with AM Fungal data.

3.4.3 **Scalability Analysis of Parallel WDA-SMACOF**

The time cost of each step in WDA-SMACOF is studied within ranging from 100k to 400k fungi sequence data, with from sequential code to 4096 processors (CPU cores) on BigRed II. The parallel applications were carried out using parallel WDA-SMACOF implemented with Harp. Figure 3.25 illustrates the result of increasing data size by fixing the number of processors, i.e. *strong scaling*. Note that the computation here is done by using one processor per core. As there are 32 cores per node, and one core can only get 2GB memory allocation, so the memory is not sufficient to run larger than 100k data on 512 processors, 200k data on 1024 processors, and 300k data on 2048 processors. By looking into detail of time cost for 4096 processors, when running for 400k data, the time cost is 3392 seconds, and it is 1.6 times larger than processing 300k data. This is because the algorithm is quadratic, so the theoretical time cost (without any overhead) for 400k data over 300k data should be (400k ^ 2) / (300k ^2) = 1.78, which is close to the actual 1.6. This result stands true also from the ratio of 300k to 200k, where the actual time cost is 2.8 times higher compared to the theoretical number as 2.3, and 200k to 100k that the actual time cost is 3.2 times higher compared to the theoretical number as 4. This along with the single core time cost of CG versus matrix inversion shown in Figure 3.25 proves that the WDA-SMACOF has a quadratic time complexity on both sequential version and parallel version.

The time cost of parallel WDA-SMACOF can be divided into three large parts: *initialization*, *communication* and *computation*. The *initialization* includes the time cost of loading the original distance matrix and weight matrix into cache memory, which is mainly file I/O. It also includes the time cost of scheduling from MapReduce. This time cost is fixed disregard the number of iterations because it only needed to be done once per run. *Communication* is mainly the broadcast time of matrix X as well as the collective communication used in Harp that reducer collects the output from mapper. Computation is the main time cost, which includes the computation of the main algorithm and formula. The detailed time cost of each step is given in Figure 3.26. This experiment increases the data size from 100k to 400k, and increases the number of cores from 256 to 4096. Each run is using the minimum number of cores. It shows that with the increasing number of processors, the computation cost proportion over the main computation is decreasing, but still

take up to 65% for 4096 cores on 128 nodes. When the data size is 100k, the computation takes up to 90% of the total computation time. The communication overhead is increasing with the number of nodes, increases from 8% on 8 nodes to 30% on 128 nodes. This means the design of parallel WDA-SMACOF is scalable, and the performance of it while increasing the data size as well as the number of processors remains reliable.

The computation of WDA-SMACOF includes three parts: first part is the calculation of B(Z)Z, which is the right hand side of equation (17); second part is the calculation of CG; last part is the STRESS calculation. In one SMACOF iteration, B(Z)Z computation is done once as well as the STRESS calculation, but the CG calculation involves multiple iterations of matrix multiplication operations. The detailed computation time of 100k data over increasing number of cores is given in Figure 3.27. This figure shows that the proportion of detailed computation time does not vary much with increasing number of processors since the parallelization of these three jobs are in same fashion. Averagely, STRESS calculation always takes lowest, around 24.2% of the total computation time, while CG takes around 32.4% of the total computation time. The most dominant part of computation is B(Z)Z for this case, as it takes averagely 43.4% of the total computation time. It proves that the parallel WDA-SMACOF computation won't be affected be the increasing number of cores. Note that the result in Figure 3.27 has a variety number, which is the CG iterations per SMACOF iteration. This is mainly because CG only takes averagely 2.18 iterations per SMACOF iterations for this 100k data case. Averagely, one B(Z)Z operation is 3.1 times longer than one CG iteration, and is 1.8 times longer than one STRESS calculation operation for 100k data over processors from 256 to 4096.

If CG takes a larger number of iterations, the proportion of time cost on CG computation would be larger. By increasing the data size, the CG iteration per SMACOF iteration could be increased slightly. But the time cost per SMACOF iteration and per CG iteration remains the same as shown in Figure 3.28. It shows is the time cost for each individual MapReduce job using 4096 processors while the data size increases from 100k to 400k. It shows that the time cost of B(Z)Z is increasing quadratic with the data size, as same as CG and STRESS calculation per iteration. Note that all time cost listed here are for single MapReduce job without iterations. And the CG still takes

averagely around 3 times longer than one CG computation and about 2 times longer than the STRESS computation.

By fixing the data size to 100k, and increases the number of processors, i.e. weak scaling, the time cost is shown in Figure 3.29. The WDA-SMACOF has been parallelized using two iterative MapReduce frameworks: Twister and Harp. Twister uses a broker to handle the communication within iterations while Harp uses collective communication techniques. These two implementations are denoted as WDA-SMACOF (Harp) and WDA-SMACOF (Twister). It shows that with the help of increasing number of processors, the time cost of Harp for 100k data has reduced from 1043 seconds to 243 seconds. And the parallel efficiency is shown in Figure 3.30. It almost near 1 on for 32 nodes (1024 processors), but decreases to 0.82 on 64 nodes (2048 processors). Finally when using 4096 cores, it decreases to 0.56. This is because by fixing the data size, the communication overhead increases with the number of processors and the computation on each processors decreases, so the parallel efficiency decreases.

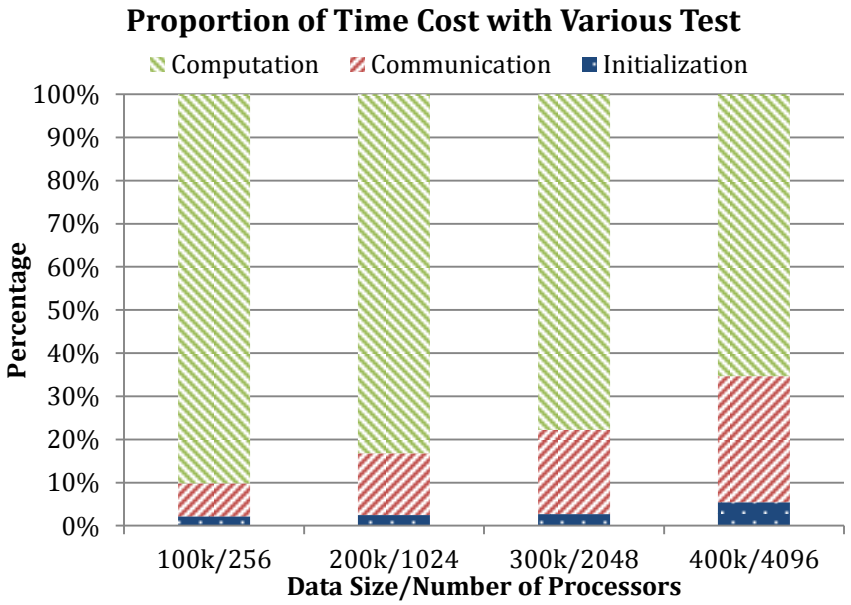

Figure 3.26: The time cost proportion each steps of parallel WDA-SMACOF. The data size and number of processors vary.

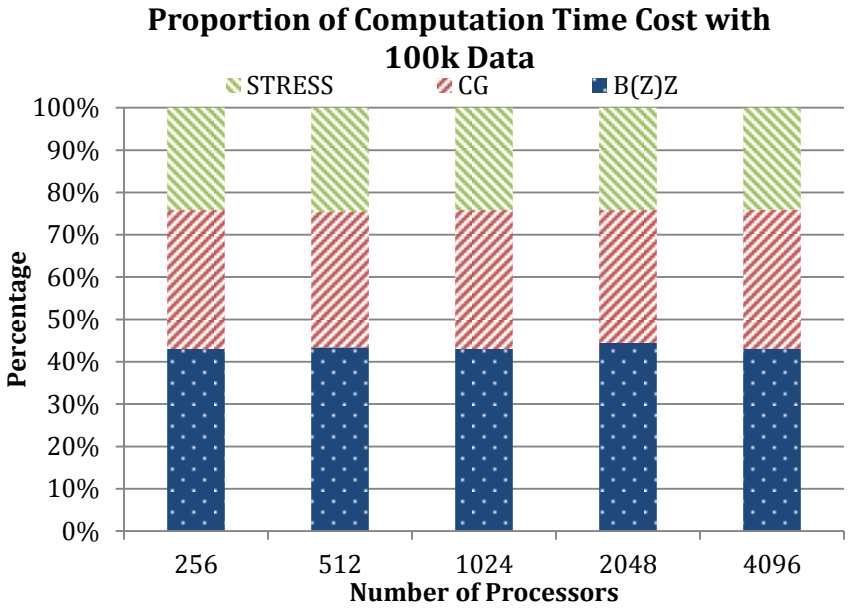

Figure 3.27: The time cost proportion in one SMACOF iteration of parallel WDA-SMACOF with 100k AM Fungal data by increasing number of processors.

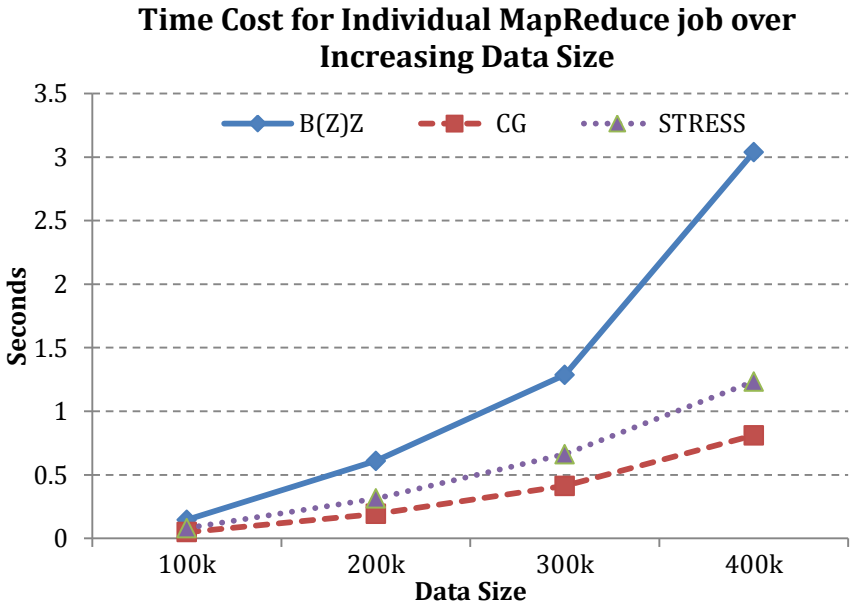

Figure 3.28: The average time cost of the single MapReduce job for three core steps in parallel WDA-SMACOF with increasing AM fungal data size.

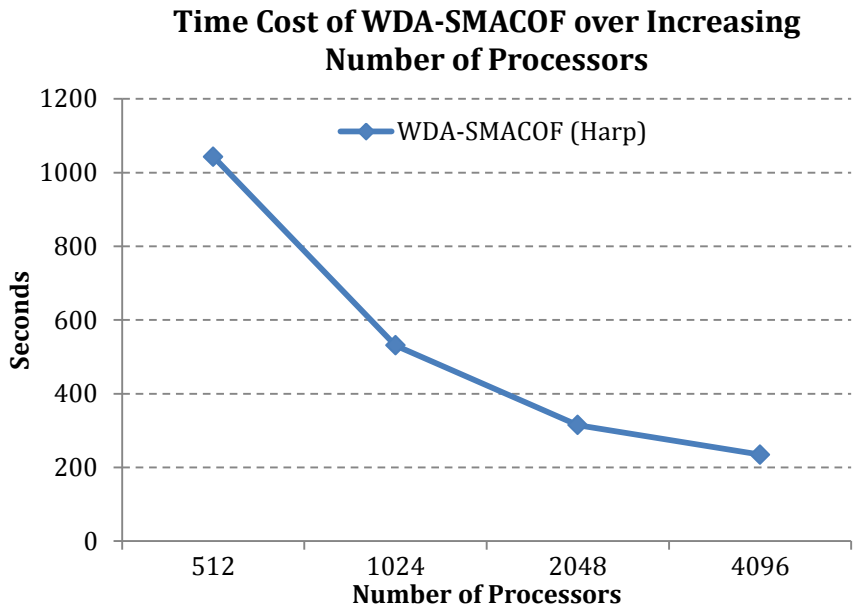

Figure 3.29: The time cost of parallel WDA-SMACOF processing 100k AM Fungal data with number of processors increased from 512 to 4096.

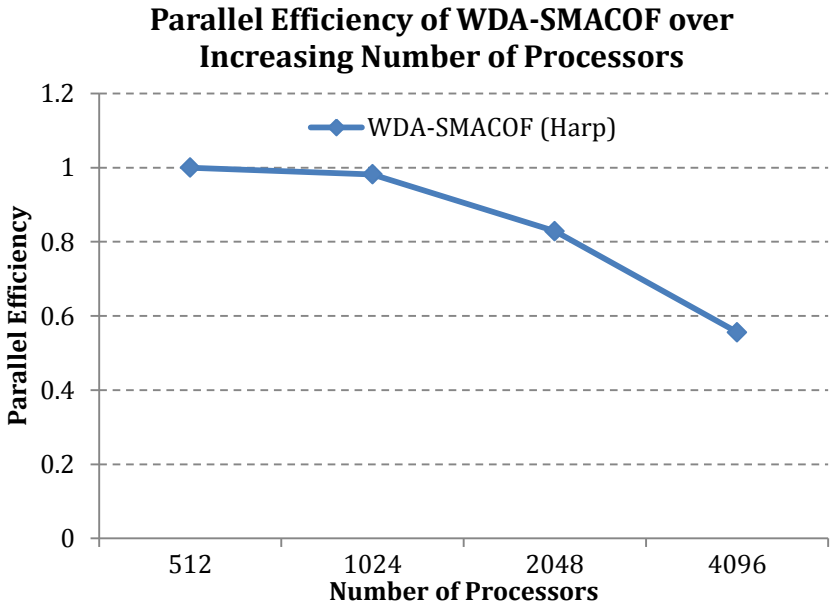

Figure 3.30: The parallel efficiency of parallel WDA-SMACOF processing 100k AM Fungal data with number of processors increased from 512 to 4096.

3.4.4 **Accuracy of Fixed-WDA-SMACOF**

The Fixed-WDA-SMACOF is compared with a normal interpolation technique called MI-MDS in order to check its time cost and accuracy. The time cost is calculated without the distance computation for the interpolation technique, i.e. MI-MDS, since distance computation is not included within the time cost of WDA-SMACOF. The accuracy is evaluated using normalized STRESS value mentioned in equation (37) for all of the data including in-sample and out-of-sample. The in-sample data were processed using WDA-SMACOF in order to get the most accurate result, then the rest out-of-sample data were either interpolated to the target dimension or being processed using fixed WDA-SMACOF. The number of nearest neighbor that used in MI-MDS is always the in-sample data size.

This experiment is carried out on the 4640 Artificial RNA data, where in-sample data size increased from 500 to 4000, and the out-of-sample data size decreases from 4140 to 640. The normalized STRESS value is given in Figure 3.31 and time cost is given in Figure 3.32. As shown in Figure 3.31, the normalized STRESS value of MI-MDS and WDA-SMACOF are both decreasing when in-sample size increases. This is because with the increasing in-sample size, the part of data that needs to be processed decreases. And the full WDA-SMACOF on the original data obviously has a lower STRESS value compared to the fixed WDA-SMACOF or interpolation techniques. On the other hand, the WDA-SMACOF is more accurate than MI-MDS because it also considered about the distances within the out-of-sample data. The differences between the their normalized STRESS decreases when the in-sample size increases as well. And when the in-sample size reaches 4k, their differences is only 0.00002, which is different from the result from in-sample size 500, where their differences is 0.0007. This means the fixed WDA-SMACOF has a much larger advantage while processing data with a small in-sample size. Figure 3.32 illustrate the time cost of both the methods, and WDA-SMACOF is much slower than the MI-MDS method. This is because WDA-SMACOF has a quadratic time complexity while MI-MDS is linear. The time cost of WDA-SMACOF decreases with the out-of-sample data size, and the time cost of MI-MDS reaches maximum as in-sample size 2000. This is because the time cost of MI-MDS is $O(N_1 \times N_2)$ where N_1 is the in-sample size and N_2 is the out-of-sample size.

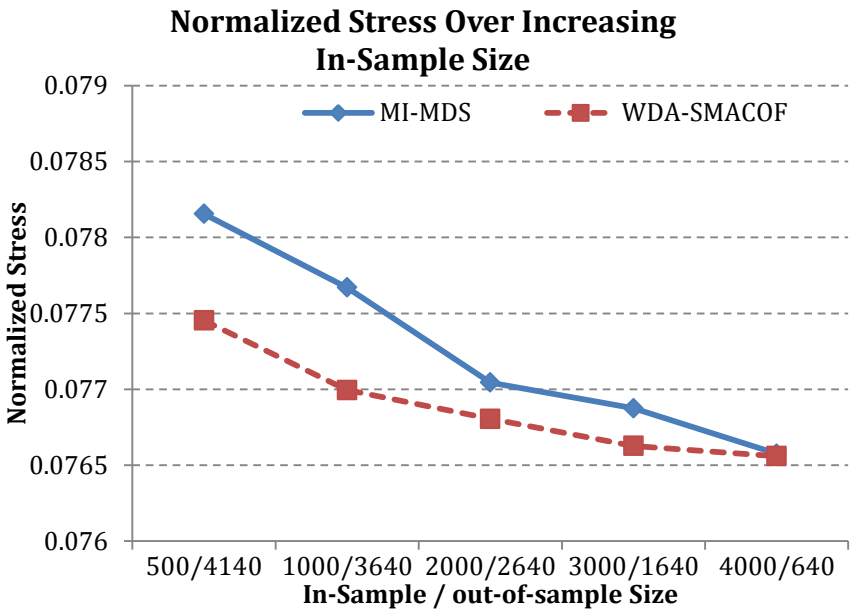

Figure 3.31: The normalized STRESS value comparison of WDA-SMACOF and MI-MDS. The in-sample data coordinates are fixed, and rest out-of-sample data coordinates can be varied.

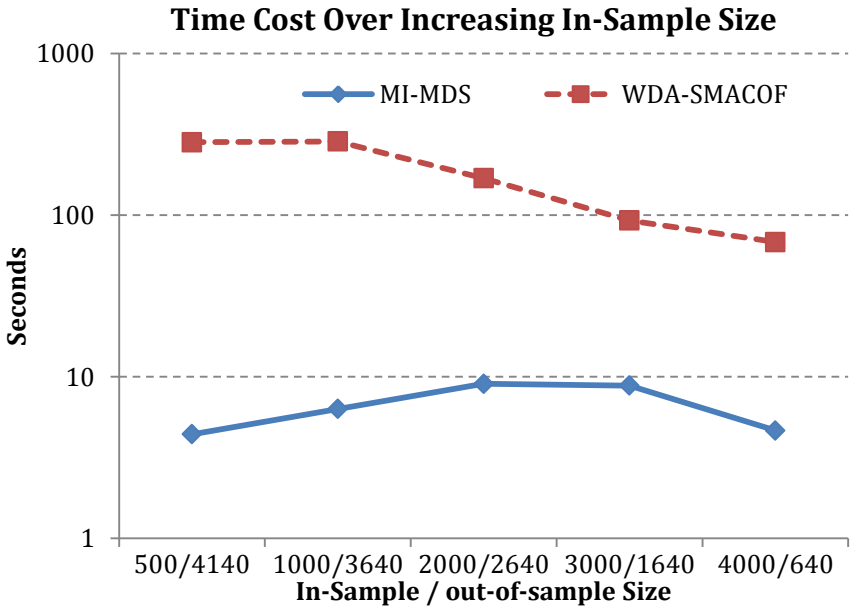

Figure 3.32: The time cost comparison of WDA-SMACOF and MI-MDS. The in-sample data coordinates are fixed, and rest out-of-sample data coordinates can be varied.

3.5 Conclusion

In this section, WDA-SMACOF and Fixed-WDA-SMACOF are proposed. WDA-SMACOF is an algorithm that can process input distance matrix with different weight associate with each distance, so that the particular needs from dissimilarities generated from pairwise sequence alignment can be met. During the performance analysis, WDA-SMACOF has been proved to have the most robust and most accurate result comparing to other existing SMACOF algorithms when applied to DNA, RNA and Protein datasets. And it has a lower time cost compare to the traditional matrix inversion solution dealing with weight function so that it can be scale to process nearly half of million data points on over 4000 processors by using iterative MapReduce framework. Fixed-WDA-SMACOF is an extension to WDA-SMACOF, so that during the dimension reduction process, the coordinates of a certain part of data points can remain fixed. This is usually for solving problems associate with reference sequences and new sequences. The experiments showed that Fixed-WDA-SMACOF has a higher accuracy than the interpolation solution by sacrificing computation time. This solution can be utilized for future work once a precise dimension reduction result needs to be obtained with fixed coordinates on a certain set of sequences in target dimension.

Chapter 4. ROBUST AND SCALABLE INTERPOLATIVE MULTIDIMENSIONAL SCALING

4.1 Overview

Interpolation [22] is a solution for processing very large scale MDS problems, such as for millions of sequences. This is because MDS requires quadratic space (memory) and time to process, and if the data size increases to very large scale and computing resources is limited, one has to use alternative solutions to address the dimension reduction problem, or known as out-of-sample problem. One common alternative is to use interpolation, For interpolation, the input dataset is split into two parts: in-sample dataset and out-of-sample dataset. The MDS algorithm is applied on the in-sample dataset, so that a high accuracy result could be generated. Then the result of the data, referred to as out-of-sample dataset, can be interpolated to the in-sample's low dimension space by using the high dimension distances generated from in-sample points to the out-of-sample points. The distances generated in high dimension are the sequence alignment dissimilarities for DACIDR. As the sequence dissimilarities are not as reliable as the Euclidean distances, so weights

may also need to be associated with different distances. This weighted solution for the interpolation is called W-MI-MDS.

Additionally, the points are independent from each other, so the algorithm could be parallelized pleasingly. It is also referred to as task-independent parallelization, and it is also referred to as Online MDS. A tradition way of splitting them is to set the in-sample set as large as possible for the MDS algorithm since it can generate a more accurate result, then interpolate the rest of the points into the target dimension. So if the in-sample data size is large, by linearly interpolating the points into the target dimension, the speed of it may not suffice the requirement as real-time processing speed. In order to process the data points with a faster speed, some hierarchical optimization has been discussed in this section in addition to the W-MI-MDS.

4.2 **Related Work**

To address the out-of-sample problem, many algorithms have been developed to extend the capability of various dimension reduction algorithms by embedding new points with respect to previously configured points. An out-of-sample [65] solution has been proposed to multiple unsupervised learning algorithms, to name a few, MDS, spectral clustering, Isomap and LLE. The solution was based on a spectral embedding of the data and it was a generalized solution that shows the embedding between in-sample and out-of-sample points is comparable due to replacing a few points in the training set. An extension for the algorithms [66] has been proposed based on the latent variable model by deriving from semi-supervise learning to define out-of-sample mappings for Laplacian eigenmaps. In MDS, the out-of-sample problem could also be considered as unfolding problem [11] since only pairwise dissimilarities between in-sample sequences and out-of-sample sequences are observed. A generalized out-of-sample solution [67] has been provided that uses coordinate propagation for non-linear dimension reduction. An out-of-sample extension [68] for the Classical Multidimensional Scaling (CMDS) has been proposed. It has applied linear discriminant analysis to the labeled objects in the representation space. In contrast to them, an EM-like optimization solution, called MI-MDS [22] is proposed to solve the problem

with STRESS criteria in equation (4), which found embedding of approximating to the distance rather than the inner product as in CMDS.

As for hierarchical method, ESPRIT-Tree [69] has been proposed address the hierarchical clustering computation time and space issue. It uses probability sequences and a tree-like structure in hyperspace to reduce the time and memory usage for sequence analysis where its tree construction relies on a subset of result from ESPRIT. Although by using ESPRIT-Tree, sequence clustering has a time complexity of O(NlogN), but the tree construction itself takes $O(N^2)$ time, which can only be applied on small dataset. Kd-tree [70] is a famous binary tree for fast data point classification, and some optimized methods [71, 72] based on it has been proposed later for the multiple dimensions on different tree level. However, this type of methods can only be applied on the situation that out-of-sample points have a high dimension coordinates as well as the in-sample points. Ball-Tree was based on the construction process of Kd-tree, and it does not need the out-of-sample point to have a coordinates in the original dimension. Cover-tree was later proposed as an optimized tree based on it. The creation of these types of tree would require the in-sample points to use its original dimensionality, which is not available for sequence clustering. And one can adapt to use the coordinates after dimension reduction to do that.

4.3 **Weighted MI-MDS**

MI-MDS is an iterative majorization algorithm that can minimize the STRESS value in equation (4), where all weights are assumed to be 1. It will find k nearest neighbors from in-sample points of a given out-of-sample point at first, and then reduce the STRESS from this out-of-sample point to its k nearest neighbors by finding the optimized coordinates for it. Then by finding a majorizing function, its minimum STRESS can be obtained analytically.

MI-MDS has been proved to be efficient when deal with large-scale data. However, it assumed that all weights equal to one, where it couldn't deal with missing values and different weights. As mentioned in last section, the dissimilarities generated with sequence alignment may be in need of assigning weights to the corresponding distances. Therefore, we propose W-MI-MDS to solve these issues. To solve the weighted out-of-sample problem, we need to find an optimization

function for weighted STRESS function for equation (3) instead of equation (4). Note that the weight added here is the corresponding weights from the sequence alignment dissimilarity between out-of-sample sequence $\hat{x}$ and in-sample point i.

4.3.1 **Algorithm**

By expanding equation (3), the STRESS function can be written in

$$\sigma(X) = \sum_{i \le N} w_{i\hat{x}} \delta_{i\hat{x}}^2 + \sum_{i \le N} w_{i\hat{x}} d_{i\hat{x}}^2(X) + 2 \sum_{i \le N} w_{i\hat{x}} \delta_{i\hat{x}} d_{i\hat{x}} \quad (38)$$

$$= \eta_\delta^2 + \eta^2(X) - 2\rho(X) \quad (39)$$

where η_δ^2 is a constant irrelevant to X. So similar to the majorizing process in SMACOF, only $\eta^2(X)$ and $\rho(X)$ need to be considered to obtain the majorization function. $\eta^2(X)$ can be deployed to

$$\eta^2(X) = w_{1\hat{x}} \|\hat{x} - p_1\|^2 + \cdots + w_{N\hat{x}} \|\hat{x} - p_N\|^2 \quad (40)$$

$$= \sum_{i \le N} w_{i\hat{x}} \|\hat{x}\|^2 + \sum_{i \le N} w_{i\hat{x}} \|p_i\|^2 - 2\hat{x}q^t \quad (41)$$

where $q^t = (\sum_{i \le N} w_{i\hat{x}} p_{i1}, \dots \sum_{i \le N} w_{i\hat{x}} p_{iL})$ and L is the target dimension. The *Cauchy-Schwarz inequality* can be applied on $-d_{i\hat{x}}$ in $\rho(X)$ to establish the majorization function. Assume the distances between two points i and j in a matrix X, which are the *i*th row and *j*th row in matrix X, denoted as x_i and x_j, and in a different matrix Z, can be denoted as z_i and z_j. According to *Cauchy-Schwarz inequality*, there is

$$\sum_{a \le L} (x_{ia} - x_{ja})(z_{ia} - z_{ja}) \le \left(\sum_{a \le L} (x_{ia} - x_{ja})^2\right)^{\frac{1}{2}} \left(\sum_{a \le L} (z_{ia} - z_{ja})^2\right)^{\frac{1}{2}} \quad (42)$$

Assume that point i is in-sample point, which remains the same, and x_j the out-of-sample point as $\hat{x}$, the equation is updated as

$$\sum_{a \le L} (p_{ia} - \hat{x}_a)(p_{ia} - z_a) \le (\sum_{a \le L} (p_{ia} - \hat{x}_a)^2)^{\frac{1}{2}} (\sum_{a \le L} (p_{ia} - z_a)^2)^{\frac{1}{2}} \quad (43)$$

$$\sum_{a \le L} (p_{ia} - \hat{x}_a)(p_{ia} - z_a) \le d_{i\hat{x}} d_{iz} \quad (44)$$

So the inequality is obtained as

$$d_{i\hat{x}} \geq \frac{\sum_{a \leq L}(p_i - \hat{x})(p_i - z)}{d_{iz}} \quad (45)$$

And finally, it is given as

$$-d_{i\hat{x}} = -\|\hat{x} - p_i\| \quad (46)$$

$$\leq -\frac{\sum_{a \leq L}(p_{ia} - \hat{x}_a)(p_{ia} - z_a)}{d_{i\hat{x}}} \quad (47)$$

$$= -\frac{(p_i - \hat{x})^t (p_i - z)}{d_{iz}} \quad (48)$$

where z is a vector of length L which contains $(z_1, \dots, z_L)$, and $d_{ix} = \|p_i - z\|$. Do the summation over all the in-sample point in P, then applying equation (48) to $\rho(X)$, we will have

$$-\rho(X) \leq -\sum_{i \leq N} w_{ix}\delta_{ix} \frac{(p_i - \hat{x})^t (p_i - z)}{d_{iz}} \quad (49)$$

$$= -\hat{x}^t \sum_{i \leq N} \frac{w_{ix}\delta_{ix}}{d_{iz}} (z - p_i) + C \quad (50)$$

where C is a constant irrelevant from x. After applying equation (41) and (50) to (39), we will have

$$\sigma(X) \leq \eta_\delta^2 + \sum_{i \leq N} w_{i\hat{x}} \|\hat{x}\|^2 + \sum_{i \leq N} w_{i\hat{x}} \|p_i\|^2 - 2\hat{x}q^t - \hat{x}^t \sum_{i \leq N} \frac{w_{i\hat{x}}\delta_{i\hat{x}}}{d_{iz}} (z - p_i) + C = \tau(\hat{x}, z) \quad (51)$$

As both η_δ^2 and C are constants, equation (51) is a majorization function of the STRESS that is quadratic in X. The minimum of this function can be obtained by setting the derivatives of $\tau(\hat{x}, z)$ to zero, that is

$$2\sum_{i \leq N} w_{i\hat{x}}\, \hat{x} - 2q^t - 2\sum_{i \leq N} \frac{w_{i\hat{x}}\delta_{i\hat{x}}}{d_{iz}} (z - p_i) = 0 \quad (52)$$

$$\hat{x}^u = \frac{q^t + \sum_{i \leq N} \frac{w_{i\hat{x}}\delta_{i\hat{x}}}{d_{iz}} (z - p_i)}{\sum_{i \leq N} w_{i\hat{x}}} \quad (53)$$

where z is the previous estimated $\hat{x}$. This formula is treated as the final formula for W-MI-MDS, which can guarantee to generate a series of non-increasing STRESS value for from original distances with various weights. One can simply apply equation (10) to this formula to add DA optimization to this algorithm, which is given as:

$$\hat{x}_t^u = \frac{q^t + \sum_{i \le N} \frac{w_{i\hat{x}} \tilde{\delta}_{i\hat{x}_t}}{d_{iz}} (z - p_i)}{\sum_{i \le N} w_{i\hat{x}}} \quad (54)$$

where $\tilde{\delta}_{ix_t}$ can be obtained using equation (10). But in practice, it is found that this algorithm is hard to be trapped under local optima. So unless absolute robustness is needed for interpolation algorithm, it is not necessary to run the D technique on W-MI-MDS.

Algorithm 3 W-MI-MDS algorithm

Input: $\boldsymbol{D_1}$, $\boldsymbol{D_2}$, $\boldsymbol{X_1}$, $\boldsymbol{\varepsilon}$ and $\boldsymbol{\alpha}$
Output: $\boldsymbol{X_2}$ as target dimension mapping

```
For each x̂ in X_2 do
    Initialize random mapping for x̂, called x̂^|0|
    while σ(x̂^|t|) − σ(x̂^|t+1|) > ε do
        Update x̂^|t+1| using (53)
        t = t + 1
    end while
end for
return X_2
```

4.3.2 **Parallelization of W-MI-MDS**

The parallelization of W-MI-MDS can be done on only MapReduce framework since it is a pleasingly parallel application. Only map tasks are needed for the computation and there are no iterations or communication between each independent map task. The flowchart is illustrated in Figure 4.1. D_1 and X_1 are copied and loaded into memory on every mapper as the space cost is linear, and D_2 is partitioned and distributed across the mappers. Only one reducer is used here to combining the final result, which will be in L-dimension so the time cost of doing that in a single thread is low.

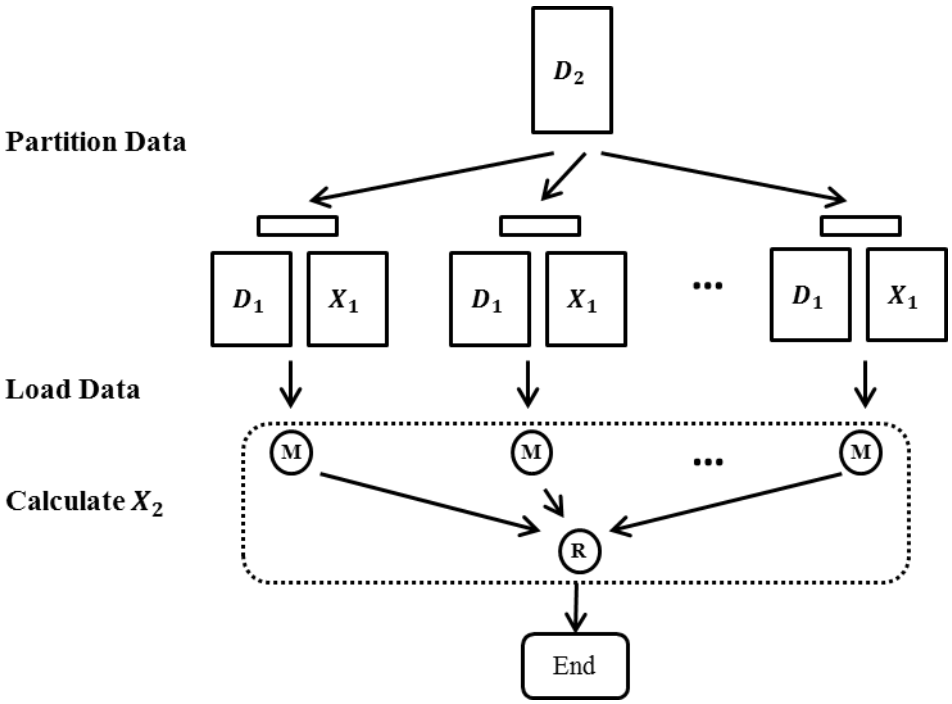

Figure 4.1: The flowchart of parallel W-MI-MDS using an MapReduce framework

4.4 Hierarchical Interpolation

The hierarchical approach is needed for W-MI-MDS is because of the slow sequence alignment speed. Since in MI-MDS, all distances needs to be calculated in order to find the k-nearest neighbors for interpolation, the distance calculation of MI-MDS is $N_1 * N_2$ assuming there are N_1 in-sample points and N_2 out-of-sample points. Therefore, each sequence in in-sample set needs to be aligned with each sequence in the out-of-sample dataset. In practice, this computation time is dominated by the sequence alignment. In our test, an ASA with 100k 16s rRNA needed several hours to finish on 800 cores, the total number of alignments in that computation is 100k * 100k / 2. If this 100k is considered as sample set and the rest one million sequences as out-of-sample set, the total number of alignments will increase to 1m * 100k, which will take 18 times longer than the ASA computation. Furthermore, for purely computation for the dimension reduction, less in-sample point always means a faster speed according to equation (54). So in order to address this

issue, two novel tree structures has been proposed in here and further performance analysis are illustrated in the following section.

4.4.1 **Sample Space Partition Approach**

One way to approach the hierarchical solution is to partition the target dimension space, referred to as sample space here, in order to generate a tree structure. The concept from astrophysics simulations (solving $O(N^2)$ particle dynamics) is used here to split the sample data in L=3-dimension space into an octree with Barnes-Hut Tree (BH-Tree) [73] techniques. The BH-Tree is used to reduce the time cost of computing nbody [74] simulation. It recursively divide the n bodies into quad-tree in 2D space or oct-tree in 3D space, then each node can represents all the bodies within the its region. Inspired by BH-Tree, the tree called Sample Sequence Partition Tree (SSP-Tree) proposed in this section is similar to that. First, the sample dataset is divided up into cubic cells via an octree (in an L=3-dimension space). where the tree node set is divided into two sets: leaf node set and internal node set. Each leaf node contains one sequence, and each internal node contains all the sequences belong to its decedents. Each internal node has a child nodes set which contains the number of its children smaller or equals to 2^L. Figure 4.2 is an example shown how the SSP-Tree works in 2D with 8 sequences. If a node contains only one sequence, then it becomes a leaf node; otherwise it is an internal node. Leaf node e0 to e7 contains the sequences from A to H accordingly. The internal node i1 contains sequences A, B, C and D. i2 contains sequences G and H, and i0 contains all the sequence as it is the biggest box.

A tree node *i* can be represented in only two points in dimension L, which are $X_i^{max} = (x_1^{max}, x_2^{max}, x_3^{max}, \dots, x_L^{max})$ and $X_i^{min} = (x_1^{min}, x_2^{min}, x_3^{min}, \dots, x_L^{min})$ where x_j^{max} denotes the maximum and x_j^{min} denotes minimum value of all the points' coordinates value in *L* dimensions with $1 \leq j \leq L$. The reason for this representation is that this known SSP-Tree node is a cubic-type of structure, that lines connecting each vertex at corner are absolutely straight. So there is no need to store information other than these two points. Thereby, this representation is most efficient for this particular SSP-Tree.

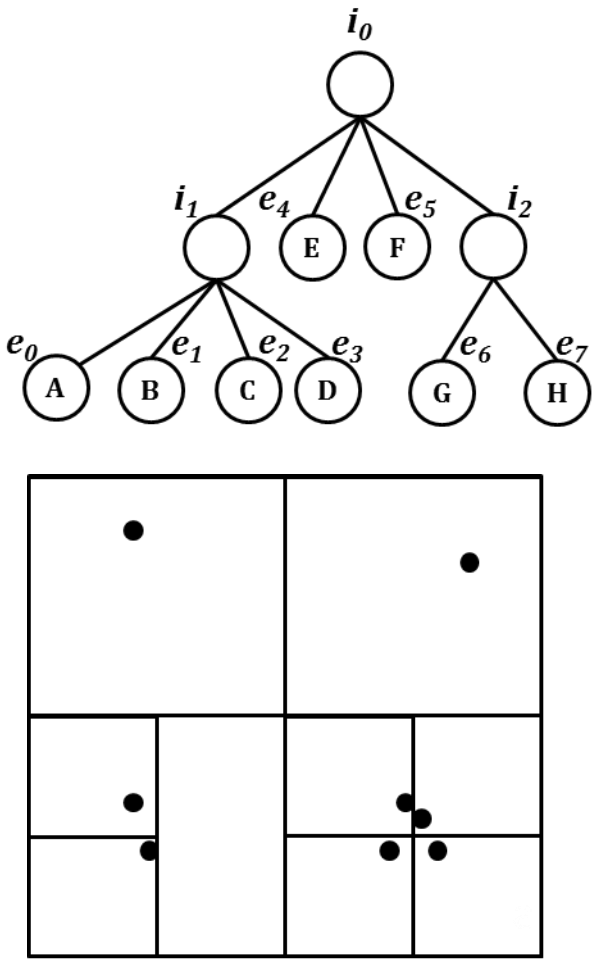

Figure 4.2: An illustration of SSP-Tree with 8 sequences. The upper chart is the tree relationships, and chart below is the actual representation of SSP-Tree in 2D.

The construction of SSP-Tree follows the next steps: First, every sample points in the target dimension is scanned, then the upper bound and lower bound of coordinates in the target dimension is found, so that the root node can be constructed, and initialized as a leaf node. Second, loop over the in-sample points, and insert the point into the SSP-Tree, following the three conditions: If current tree node has not been assigned with a sequence (in-sample point), assign the current point to it; Or if the current tree node is an internal node, insert the current node to its corresponding children determined by its geometric center; Or if the current tree node is a leaf node (already got a sequence assigned with it), then remove the sequence assigned and insert this sequence along with the new in-sample point into this leaf node again. Note that the sequence

mentioned here means the original sequence associated with the in-sample point that represents it in the target dimension space. And it is also worth mention that the only computation for it is to calculate the center of each tree node. Inserting the sample points into the tree only needs comparison and assignment. In practice, inserting a hundred thousand points into a SSP-Tree only takes about a few seconds on a desktop.

In SSP-Tree, every tree node *i* has contains a set of points, which is denoted as *Pi*. After generate the tree from the in-sample points, there are several ways to represent the tree node with a single point inside. One way is to find the nearest point to its geometric center denoted as X_i^{mid}, which can be given as

$$X_i^{mid} = \frac{(X_i^{max} + X_i^{min})}{2} \quad (55)$$

where X_i^{max} and X_i^{min} is the representation of tree node *i*.

But in practice, the cluster of points might be away from a tree node geometric center. And this may cause the point at the edge of the cluster inside a tree node to represent the tree node instead of the point near the cluster center. So to solve that, another way of using a single point to represent the tree node is to find the point near the center of mass instead of geometric center. The mass center p_c of node *i* is given by the following equation:

$$p_c^i = \{x_l^{\mathrm{i}} \mid x_l^{\mathrm{i}} = \sum_{i=0}^{n_{\mathrm{i}}} \frac{x_l^i}{n_{\mathrm{i}}}, 0 \le l < L\} \quad (56)$$

where n_i is the number of sequences (in-sample points) in node i.

By using a single point to represent the tree node, a simple hierarchical majorizing interpolation method can works as follows: a simple hierarchical majorizing interpolation method (HI-MI) as follows: One compares an out-of-sample point $\hat{\mathrm{p}} \in \hat{\mathrm{P}}$ to representative point of tree root first, and then recursively assign $\hat{\mathrm{p}}$ to a nearest child node by comparing the distances from it to all the representative points from its children if there is any, until the node containing nearest *k* neighbors is reached. This hierarchical method can reduce the time cost of interpolation from O(N*M) to O(M*logN). However, its accuracy is poor due to the following two reasons: 1) correctness of

center point representation: it is obvious that the nodes in leaf set are represented directly by the points they contain since they only have one point per node, so the representation is 100% accurate. But their parents may contain multiple points, where could be in a same cluster or different clusters. There is a good chance that a single cluster being split to multiple internal nodes until a very high level of tree node is reached. (sometimes to the tree root). Therefore, the lower node level is, the more likely the points in that node belong to a same cluster. At upper level, the representation precision becomes worse because the points might be in different clusters. Since this method searches the tree from top to bottom, where it starts with worst representative point, there is a high probability that $\hat{p}$ could be assigned to a node rather than the node its k nearest neighbors are in. 2) The SSP-Tree is generated using distances after dimension reduction, so the distances must follows the property of Euclidean distances. However, as the out-of-sample point interpolated here is does not have a coordinates in the target dimension, the only information here is the distances from the out-of-sample points to the in-sample points. As mentioned in previous section, the distances are the dissimilarities generated from sequence alignment, and it is not as reliable as Euclidean distances. So by using the distance from high dimension to compare and find the tree node it belongs to in target dimension space may cause some bias.

4.4.2 **Hyperspace Approach**

As mentioned before, the SSP-Tree may have a problem with using distances from high dimension space from to interpolate the out-of-sample point but use the distances from target dimension space to generate the tree. So to solve this issue, one could use the distances from high dimension space to generate a tree. This idea is originally inspired by BIRCH [75] clustering, where this algorithm generates a tree in hyperspace for all the data points in order to achieve quasilinear time complexity. As this tree is in the hyperspace (there is not coordinates information for the in-sample sequences), the tree is constructed based on the closed neighborhood information, as referred to as Closest Neighbor Tree (CN-Tree).

The tree nodes of CN-Tree are divided into two sets, the leaf node set and the internal nodes set. Each leaf node contains one or more sequence, and the internal node contains all the sequences within its children. Unlike SSP-Tree, each internal node may have more than 2^L children, where its

child nodes number depends purely on the distribution of the points inside it. Note that a point can only belong to one tree node at a time, and for CN-Tree, the tree node may overlap each other. If a point lies within the radius of multiple tree node, it will always goes to the first tree node that it sees. The illustration of a CN-Tree in 2D is shown in Figure 4.3. The sequences are contained in the leaf node from e0 to e4 as this is the representation for 8 sequences. And internal node i_0 is act as the root, where i_1 and i_2 are the two large child node it contains. i_1 has 3 child node and i_2 has 2 child nodes respectively. And one point is within the radius in both i1 and i2, but it only considered as in i1 to avoid confliction. And note that each tree node in CN-Tree has an in-sample point lies on the center and each leaf node may contains multiple sequences. The leaf nodes e_0, e_3 and e_4 all contains 2 sequences while e_1 and e_2 only contains 1 sequence. This makes i_1 contains 4 sequences and i_2 contains 4 sequences in total.

In formal definition, each tree node in a CN-Tree can be denoted as $\{p_c, r\}$ where p_c is the center of that tree node, and r is the radius of the tree node. The radius can be chosen arbitrarily or depends on the overall points locations. Note that scanning over all the distances between each pair of in-sample points is very in-efficient since it has N^2 distances if there are N in-sample points. A few random distances from out-of-sample points to in-sample points can be chosen to determine the maximum radius for the root node. To generate a CN-Tree, the following steps need to be taken: First, determine the radius for each level of tree node and the maximum level for the tree; Second, for each in-sample point, compare it with the existing tree node center points within same level with a same parent, and follows the 2 situations: if it is inside the radius in an existing tree node, assign it to that tree node than insert it to the lower level if it is not the lowest level allowed; if it doesn't belongs to any tree node, create a new tree node and make it center, assign the corresponding sequence to that tree node. The difference from the construction of tree process shows the difference between CN-Tree and SSP-Tree vividly. The CN-Tree will stop when maximum level of tree node is reached (or some other termination condition, such as the smallest radius for a tree node). But SSP-Tree will only stop if every leaf node contains only one sequence, so in CN-Tree, one leaf node may contain multiple sequences. But the leaf nodes in CN-Tree usually have a very small radius in practice.

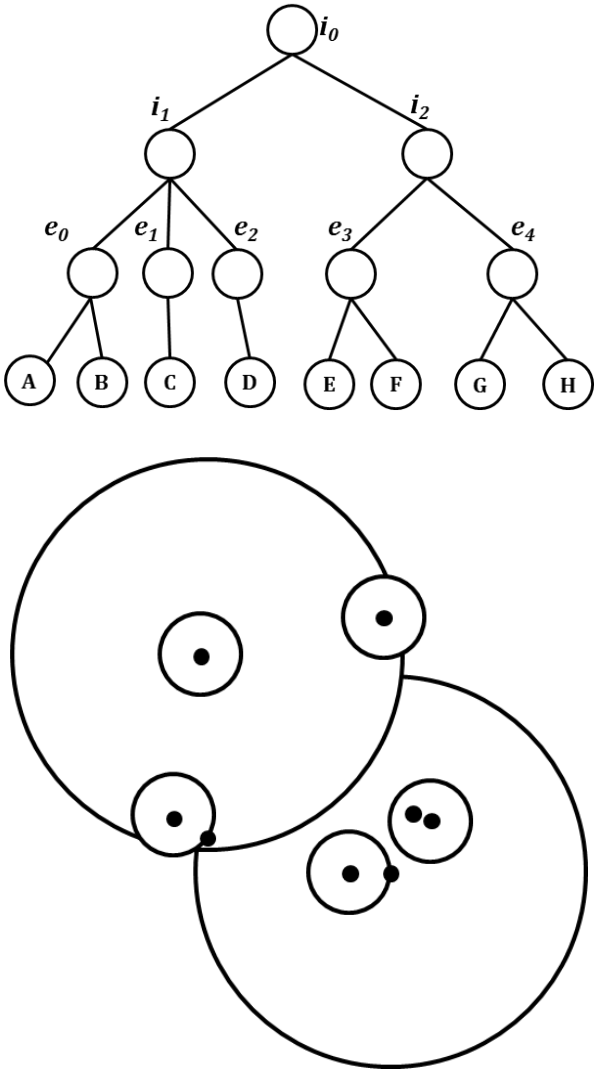

Figure 4.3: An illustration of CN-Tree with 8 sequences. The upper chart is the tree relationships, and chart below is the actual representation of SSP-Tree in hyperspace and projected to 2D.

Naturally, the CN-Tree has a point representing its center, so the out-of-sample point can directly compare to its center in order to decide the nearest tree node it finds, or being contained in. So a naïve hierarchical approach works similar to the one mentioned in SSP-Tree, where an out-of-sample point compares the distances from it to the center points of all the children from tree root, and then recursively assign $\hat{p}$ to a nearest child node by comparing the distances from it to all the representative points from its children if there is any, until the node containing nearest *k* neighbors

is reached. This approach avoids the conflicts from using high dimension distances to interpolate the point but use target dimension distances to generate the tree as CN-Tree is generated using the distances from original sequence alignment dissimilarity. However, it still suffers from the center point representation problem mentioned in SSP-Tree approach. Furthermore, as one point may belongs to a radius from multiple nodes in the same level, two points from a same cluster may belongs to two different child node because of this. And the dissimilarities from sequence alignment are not as reliable as Euclidean distances, so the CN-Tree constructed is not as reliable as SSP-Tree. Here is an example for some detailed explanation: sequence A and B are centers of two CN-Tree nodes, denote as {A, r} and {B, r}. Sequence C and D belongs to {A, r}, and if a sequence E is inserted into the tree, it may has a smaller distance with B but larger distances with A while it is very close to C and D. So if will be inserted into tree node B instead of A to construct a new child node, while it should stays with C and D and be put into a same node.

4.4.3 **Heuristic Majorizing Interpolation**

Both of the tree structures proposed here have their own advantages and disadvantages. To overcome this issue while keeping the lower time cost, a heuristic majorizing interpolation method is proposed to solve the issues mentioned above. This method puts a preprocessing of the tree generated, and instead of interpolate the out-of-sample point from the top of the tree, it will interpolate the point from a middle level of the tree, then searches for several levels until satisfied, and the lowest level of nodes are called terminal nodes. In this way, the lowest quality of representative point in both SSP-Tree and CN-Tree can be avoided. Furthermore, because of bias brought by the space partition of the tree, this approach can void that by preprocessing to find the high quality tree node first.

In formal definition, denote the terminal nodes set from both tree as T, given a terminal node $t \in T$, the number of points inside that node is denoted as N_t. The level of that tree node is given as Lt. (Note that this Lt is not the number of target dimensionality). The radius of tree node *t* is denoted as R_t, where for CN-Tree, R_t is fixed, and for SSP-Tree, R_t is given by the average over the geometric center from that tree node its furthest edge and to its shortest edge. The volume of node t is given as V_t, and the density Dt of node t is given as the following:

$$D_t = {}^{N_t}/_{V_t} \quad (57)$$

And by giving all those information of a tree node, one can derive a function, which will determine if a terminal node T is a high quality tree node or not. The general rule is to find a terminal node which contains major points from one or more clusters. And try to avoid nodes that contains some points from the edge of several clusters. Suppose all the information contains in a terminal node t can be written as a vector $\chi_t = [N_t, L_t, R_t, V_t, D_t]$. The function can be written as:

$$f(t) = \theta' \chi_t \quad (58)$$

where Θ is a vector of weights associate with each parameters in χt. The Θ can be obtained using empirical learning or use supervised learning technique such as linear regression or logistic regression. In this dissertation, the coefficient for f(t) is found using empirical experiences. A threshold, namely ε can be set to determine if node t is a high quality node or not. An example of the high quality terminal nodes found in a 3D SSP-Tree using 100k AM Fungal data is shown in Figure 4.4.

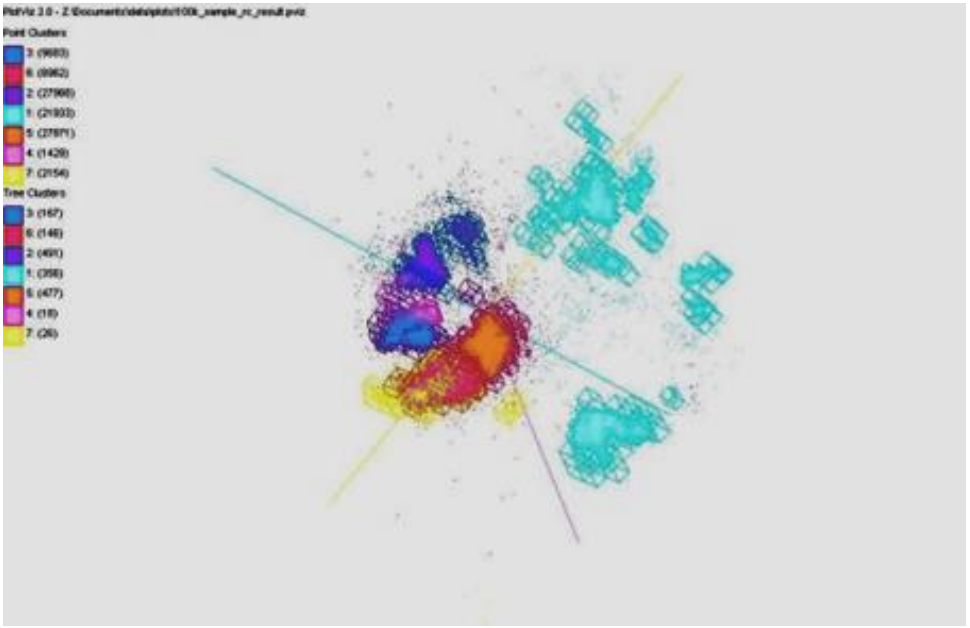

Figure 4.4: The terminal nodes generated for 100k AM Fungal data in 3D from SSP-Tree using HE-MI algorithm. The different colors represents different mega regions.

The algorithm of heuristic majorizing interpolation (HE-MI) is given in algorithm 4. In order to do the search hierarchically and try to obtain as many nearest neighbors as possible, the algorithm must consider the following situations as illustrated in Figure 4.5 to Figure 4.7. Note that these figures are the example shown as an SSP-Tree in 2D, but for other dimensions and in CN-Tree, these general rules mentioned as the following also apply. Denote the distance from an out-of-sample point to center point of tree node t as δ, and the radius of that tree node as R_t

Figure 4.5 shows that if $\delta \leq \frac{R_t}{2}$, that means the $\hat{p}$ must be inside this node, and the nearest point to it must be inside this node, too. In Figure 4.6, the situation is $\frac{R_t}{2} < \delta \leq R_t$, that means the $\hat{p}$ must be inside this node, and the nearest k points to it might be outside this node and in this node's neighbor, so one also needs to check on the neighbor of tree node t. Similar rules applied when $\delta > R_t$ as shown in Figure 4.7. Theoretically this would never happen if the distances are Euclidean distances. However, since the distances in high dimension are sequence alignment dissimilarities, this might happen and once it does, one can check the neighbors of tree node t as well to see if any tree node that could contain the nearest neighbors of $\hat{p}$ can be found.

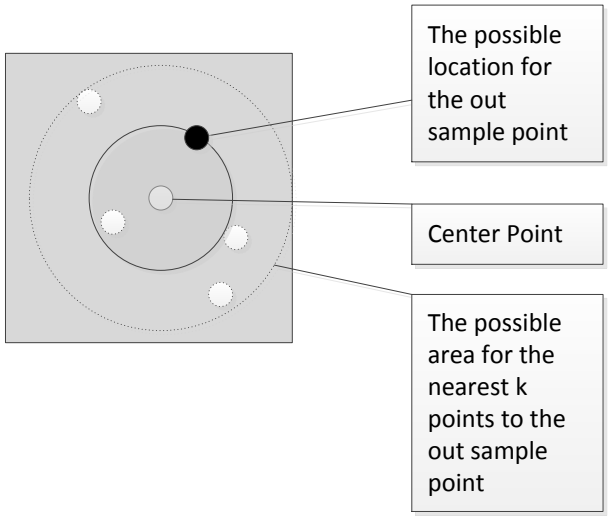

Figure 4.5: The 2D example for interpolating an out-of-sample point into in-sample space with SSP-Tree. The white points are in-sample points and the black point is the out-of-sample points. The black circle means the possible position of the out-of-sample point and the dashed circle is the possible area for the nearest neighbors of that out-of-sample point.

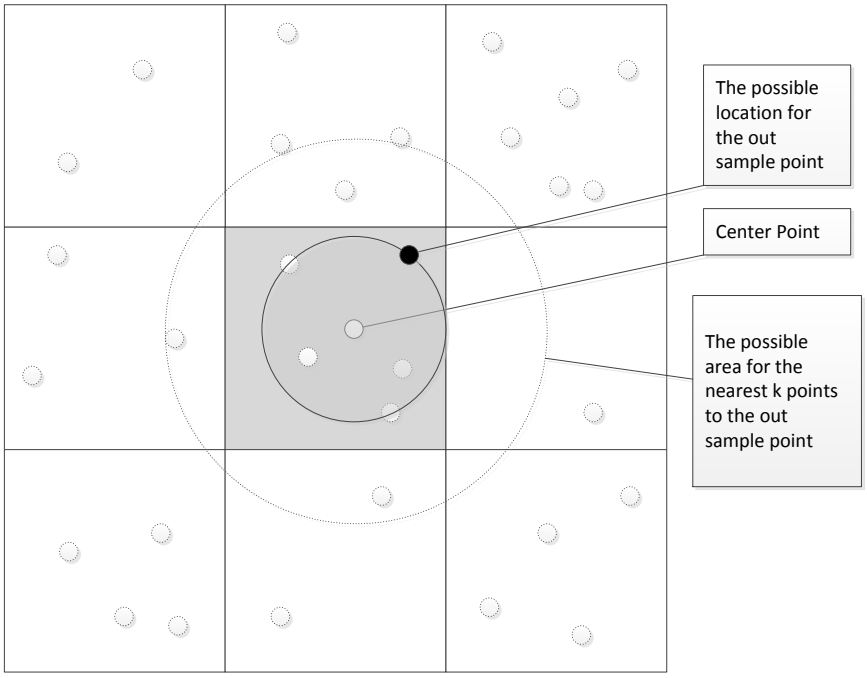

Figure 4.6: The 2D example for interpolating an out-of-sample point into in-sample space with SSP-Tree. The white points are in-sample points and the black point is the out-of-sample points. The black circle means the possible position of the out-of-sample point and the dashed circle is the possible area for the nearest neighbors of that out-of-sample point.

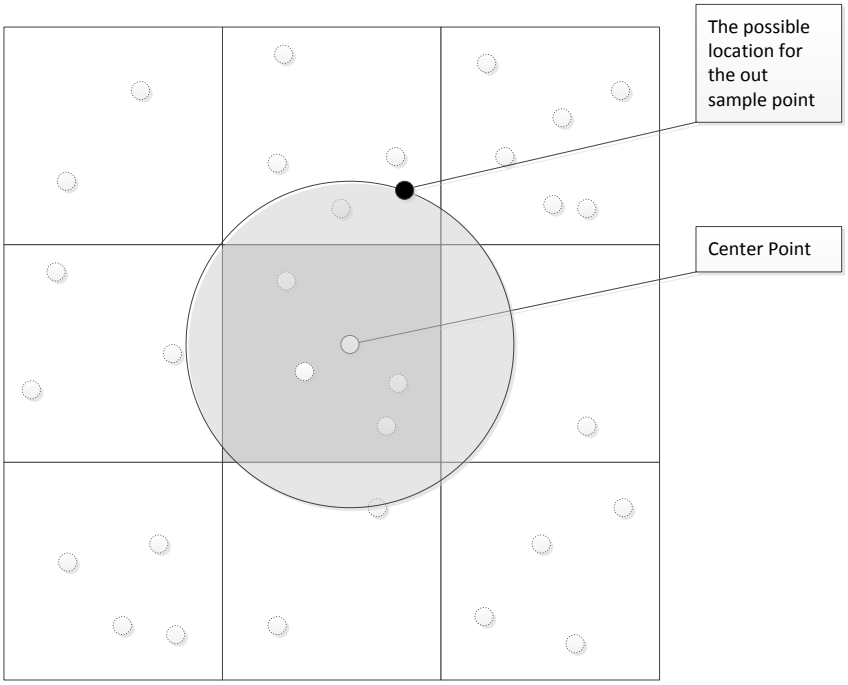

Figure 4.7: The 2D example for interpolating an out-of-sample point into in-sample space with SSP-Tree. The white points are in-sample points and the black point is the out-of-sample points. The black circle means the possible position of the out-of-sample point and the dashed circle is the possible area for the nearest neighbors of that out-of-sample point.

Algorithm 4 HE-MI algorithm

Input: P, $\hat{\mathbf{P}}$, threshold $\boldsymbol{\varepsilon}$ and N'
Output: Out-of-sample points in target dimension

```
1.  Generate SSP-Tree or CN-Tree based on P.
2.  Generate a set of terminal nodes T.
3.  For each terminal node t ∈T
4.      if f(t) < ε
5.          discard t from T
6.  End loop
7.  For each out-of-sample point p̂ ∈ P̂
8.      Function (p̂, T):
9.      Calculate the distance δ between p̂ with each p_c^t representing t ∈T
10.     find nearest terminal node denoted as t_0
11.     Create empty tree node set as T_p̂
12.     If δ < R_t / 2
13.         Add t_0 to T_p̂
14.     Else
15.         For each neighbor node t of t_0
16.             Calculate the distance δ between p̂ with p_c^t
17.             If δ < Rt
18.                 Add t to T_p̂
19.             End loop
20.             If total number of points in T_p̂ < N'
21.                 Find k nearest points from T_p̂
22.                 Apply W-MI-MDS on that
23.                 Return dimension reduction result for p̂
24.             Else
25.                 Return Function (p̂, T_p̂ )
26.     End loop;
27. Return result of P̂
```

So instead of searching through top to bottom, we can directly use the high quality representative points from high quality terminal nodes. Additionally, the number of terminal nodes is much smaller than number of in-sample points. So the time cost of HE-MI is much lower than MI-MDS which needs to compare all the sample sequences. HE-MI is described in Algorithm 4. By applying HE-MI, the time complexity is $O(M*N_T)$. The time cost is greater than the naïve hierarchical method, but the accuracy of interpolation is much higher in practice.

4.4.4 Parallelization of HE-MI

The parallelization of HE-MI is almost as same as shown in W-MI-MDS as shown in Figure 4.1. It is also a pleasingly parallel application because every out-of-sample point is independent from any other out-of-sample points. The tree constructed with in-sample points has a copy across every mapper because the memory cost if very low. And the tree construction only took a few seconds

on a single thread so that no parallelization on this part is needed. Therefore tree construction is done within each mapper independently.

4.5 Performance Analysis

In the performance analysis, the W-MI-MDS algorithm and HE-MI has been tested separately. The computing resources includes the computing cluster of BigRed II, FutureGrid Xray and Quarry. Each compute node on Quarry has 8 cores, and 16GB of memory. And BigRed II and Futuregrid Xray has been described under section 3.4. The comparison is divided into two main categories, one is to examine the accuracy and time cost of W-MI-MDS with data that has missing values; another is to do comparison of the time cost and accuracy of HE-MI using two hierarchical methods, SSP-Tree and CN-Tree with the original MI-MDS method. In Section 4.5.1 and 4.5.2, W-MI-MDS is evaluated, and in Section 4.5.3, HE-MI method is evaluated.

4.5.1 Accuracy Comparison of W-MI-MDS

Table 4 The list of data being used in following experiments

	Artificial RNA	**Hmp16SrRNA**	**COG Protein**
Number of In-sample Sequences	2000	10k	4872
Number of Out-of-sample Sequences	4640	40k	98572

In this section, Artificial RNA, hmp16SrRNA and COG Protein dataset has been used in the experiments. The size of in-sample and out-of-sample dataset of these three dataset is illustrated in Table 4. All the in-sample dimension reduction results were generated using WDA-SMACOF, as it has the lowest normalized STRESS value. Twister is acted as the MapReduce framework in these tests. For accuracy test, 4 algorithms has been tested as listed in Table 5. Note that the 2 algorithms with weighting function are derived as in Section 4.3 and the other 2 algorithms without weighting function are from MI-MDS. The DA technique has been added to both W-MI-MDS and MI-MDS, and denoted as WDA-MI-MDS and NDA-MI-MDS. The ones without DA optimization are denoted as WEM-MI-MDS and NEM-MI-MDS, which correspond to the original name W-MI-MDS and MI-MDS.

Table 5 The list of algorithms used for comparison in the performance analysis

	DA	**EM**
Weight	WDA-MI-MDS	WEM-MI-MDS
Non-Weight	NDA-MI-MDS	NEM-MI-MDS

As shown in Section 3.5, the three dataset has very different characteristics, so that the normalized STRESS value may be very different. The normalized STRESS value is given in equation (37). Each of the tests in this section includes 20 runs for each algorithm, and the error bars are the maximum value and minimum value from the runs. In all experiment within the section, the numbers of nearest neighbors were set to the maximum number of in-sample points in the same dataset. The threshold of difference between two contiguous iterations is set to 10^{-6}.

The 2640 out-of-sample Artificial RNA data were tested on a single core use sequential implementation. The 2000 in-sample data set was generated using the sequential WDA-SMACOF. 10.8% distances from the in-sample data is missing and 10.4% distances from the out-of-sample data to in-sample data is missing. The normalized STRESS comparison chart is shown in Figure 4.8. The average STRESS value for WDA-MI-MDS is 0.0423 and for NDA-MI-MDS is 0.0618. Both of the WDA-MI-MDS and WEM-MI-MDS perform 31.5% better than NDA-MI-MDS. But it shows that the DA-technique is only slight better than the original EM method. Generally speaking, the weighting function support added here is much better if there are missing distances presented.

The hmp16SrRNA out-of-sample data has a total number of 40k sequences. The in-sample dataset is generated using 80 cores on Futuregrid Xray. The rest data were interpolated using the same number of cores, where the distances were generated beforehand. were tested on a single core use sequential implementation. 10% from both the in-sample dataset and out-of-sample dataset were missing. The normalized STRESS comparison chart is shown in Figure 4.9. The performance of this shows a similar trend as in Artificial RNA data test with on a single core. Both of the WDA-MI-MDS and WEM-MI-MDS perform 33.8% better than NDA-MI-MDS. But it shows that the DA-technique is only slight better than the original EM method. So it can be safely assumed that

the parallelized version of W-MI-MDS also performs better than original MI-MDS when there are missing data presented.

The COG Protein data set is slightly different from previous two dataset. First, the in-sample data is the consensus sequences, instead of randomly chosen from the same dataset. Second, the visualization result from the dataset shows that no clear separated clusters could be observed. This is the direct reason for the higher normalized STRESS value calculated using this dataset. 10% of distances from in-sample sequences to out-of-sample sequences were missing, and 10% of distances from in-sample dataset were missing as well. The normalized STRESS value is shown in Figure 4.10. The result shows that the average normalized STRESS value is 0.1067 for WDA-MI-MDS. The non-weighted version of MI-MDS gives a normalized STRESS value of 0.1132. The WDA-MI-MDS performs 3% better than the WEM-MI-MDS, which is already a better performance compare to the performance in other dataset. The WEM-MI-MDS only has a upper boundary and lower boundary of 5*E06, which is 0.01% difference during all 20 runs in the experiments.

In all the experiments carried out for MDS interpolation, there is not much difference in all the runs in terms of final normalized STRESS value. Different from SMACOF algorithm, the normalized STRESS value from MI-MDS (NEM-MI-MDS) and W-MI-MDS (WEM-MI-MDS) won't vary much among all experiments for these very different 3 dataset. By adding DA optimization to these two algorithm, the normalized STRESS value dropped at max of 3%, with a trade off of time increasing by at least 2 times. So generally speaking, the performance of W-MI-MDS should suffice in most cases, unless very accurate result is needed.

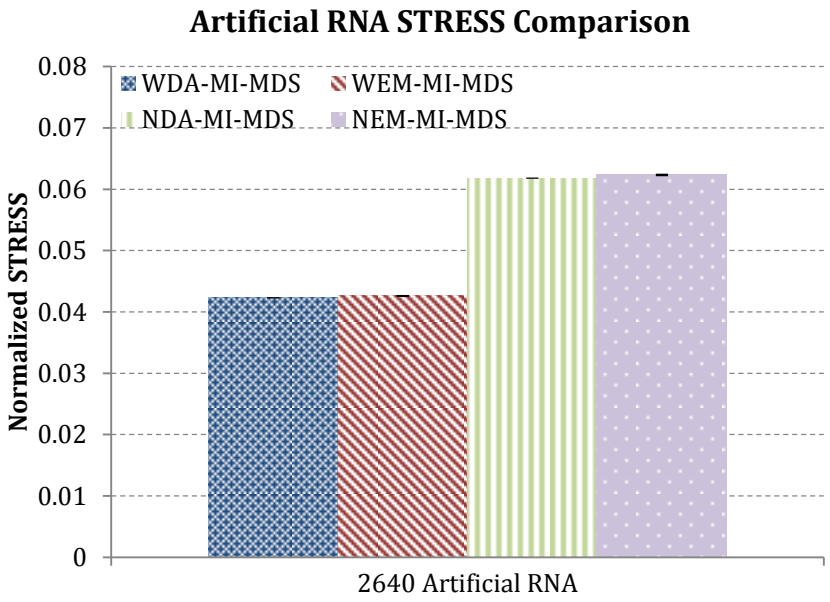

Figure 4.8: The normalized STRESS value comparison between 4 MDS algorithms using 2000 Artificial RNA sequences as in-sample data, and 2640 Artificial RNA sequences as out-of-sample data. All 4 algorithms in this experiment are sequential.

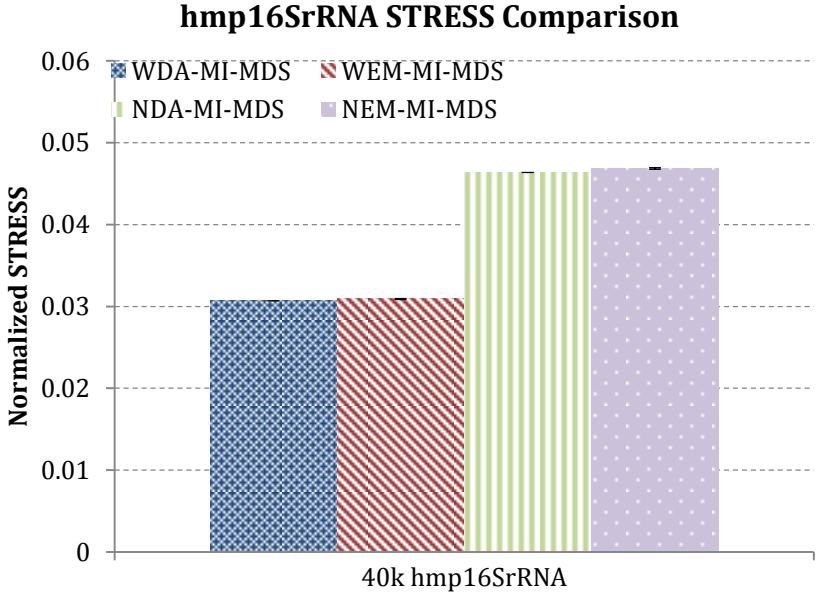

Figure 4.9: The normalized STRESS value comparison between 4 MDS algorithms using 10k hmp16SrRNA sequences as in-sample data, and 40k hmp16SrRNA sequences as out-of-sample data. All 4 algorithms in this experiment are parallel algorithms using 80 cores.

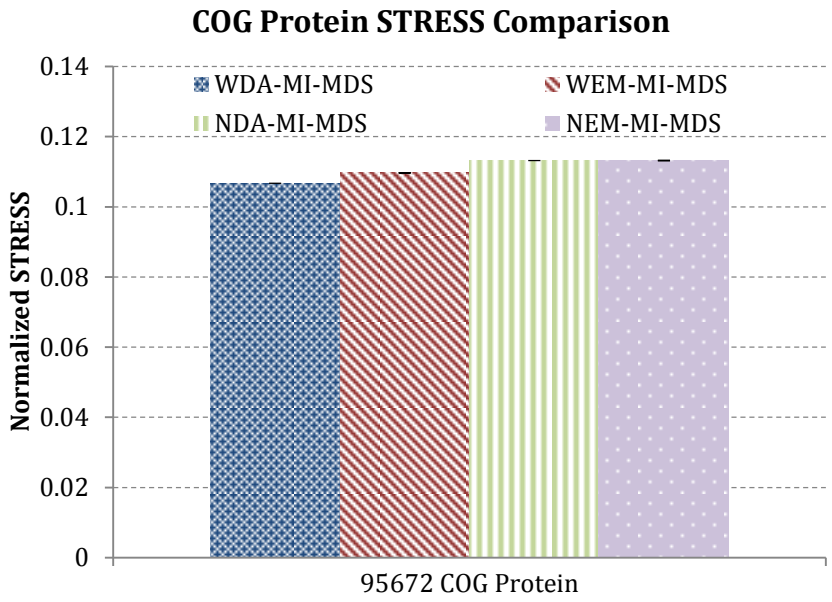

Figure 4.10: The normalized STRESS value comparison between 4 MDS algorithms using 4872 consensus sequences as in-sample data, and 95672 COG sequences as out-of-sample data. All 4 algorithms in this experiment are parallel algorithms using 40 cores.

4.5.2 **Time Cost Analysis of W-MI-MDS**

The time cost analysis for these 4 algorithms were only depends on the algorithm processing speed, since the distance calculation for each dataset is fixed and separate. In this section, the time cost of the 4 algorithm mentioned in previous section is illustrated and discussed, followed by the detailed analysis.

The time cost of interpolating 2640 out-of-sample sequences into 2000 in-sample sequences is shown in Figure 4.11. The time variance (maximum time – minimum time) of WDA-MI-MDS is averagely 11.4% of the average time cost, and for WEM-MI-MDS, it is 25.1%. So the DA technique makes the time cost more stable, although the absolute difference from maximum time cost to minimum time cost is higher. This is because the DA technique makes the WDA-MI-MDS performs 9.1 times slower than WEM-MI-MDS and NDA-MI-MDS performs 9.7 times slower than NEM-MI-MDS. Furthermore, both weighted solutions perform slight faster than non-weighted solutions.

The experiments on parallel version of these 4 algorithms shows similar result using hmp16SrRNA data as illustrated in Figure 4.12. The algorithms optimized by DA techniques are averagely 5 times slower than the original algorithm. And the algorithms using W-MI-MDS are averagely 6.3% faster than the non-weighted versions, with 10% of the distances missing in this dataset.

Finally, with the COG protein dataset, that has over 95k sequences needs to be interpolated, the result is slightly different from the previous two dataset as shown in Figure 4.13. This is due to the same reason of convergence speed mentioned in experiments carried out in Section 3.5. For this particular dataset, the non-weighted methods will not converge on the STRESS value with missing distances, rather than converge on the all the distances. So when the non-weighted program terminates, the weighted program will keep running until correct converge point is found. So the weight solution are slower because of more iterations were carried out. Also, in this test, the DA technique enabled methods only performs 1.26 times slower than the original methods. This

means for this particular dataset, where points are mostly evenly distributed over the space, the EM methods likes to converge at the same speed even with the DA optimization.

To do the future analysis on the time cost with this bias result on different dataset, the experiments were carried out with fixed number of iterations, so the time cost on each iteration can be studied. The result is shown in Figure 4.14. The dataset is selected using the 40k out-of-sample interpolated to 10k in-sample from hmp16SrRNA dataset. The iterations were fixed to 50, Note that the difference on the time cost of normal interpolation algorithms and DA optimized interpolation algorithms is the number of iterations. So to make the comparison consistent, only original algorithms were tested, denoted as W-MI-MDS for weighted version and MI-MDS for non-weighted version, which are equivalent to WDA-MI-MDS and NDA-MI-MDS with the initial temperature set to zero. The result shows that the speedup over the percentage of missing distances from W-MI-MDS to MI-MDS is almost linear. This is because the W-MI-MDS algorithm is pleasingly parallel algorithm, so there are no communications overhead. And if a distance is considered as missing, the algorithm can skip the whole computation about that in-sample point out. Therefore, when the percentage of missing distances hit 90%, W-MI-MDS is 4.15 times faster than MI-MDS.

In general, the time cost analysis shows that the W-MI-MDS can be faster than original MI-MDS when the data size is fixed. The time cost of using DA technique is generally very high. With the very little increases in accuracy and robustness, it is not suggested to apply DA technique on the interpolation unless very accurate result is needed.

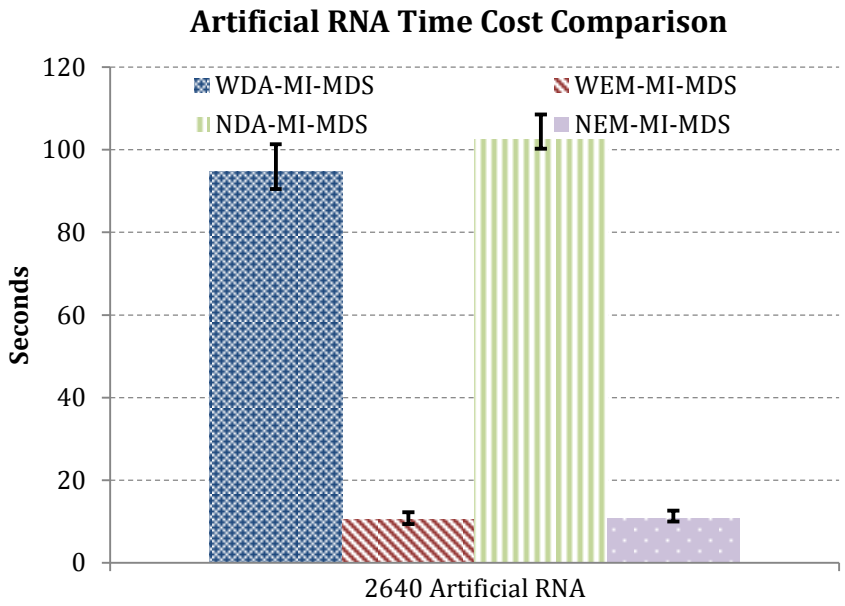

Figure 4.11: The time cost comparison between 4 MDS algorithms using 2000 Artificial RNA sequences as in-sample data, and 2640 Artificial RNA sequences as out-of-sample data. All 4 algorithms in this experiment are sequential.

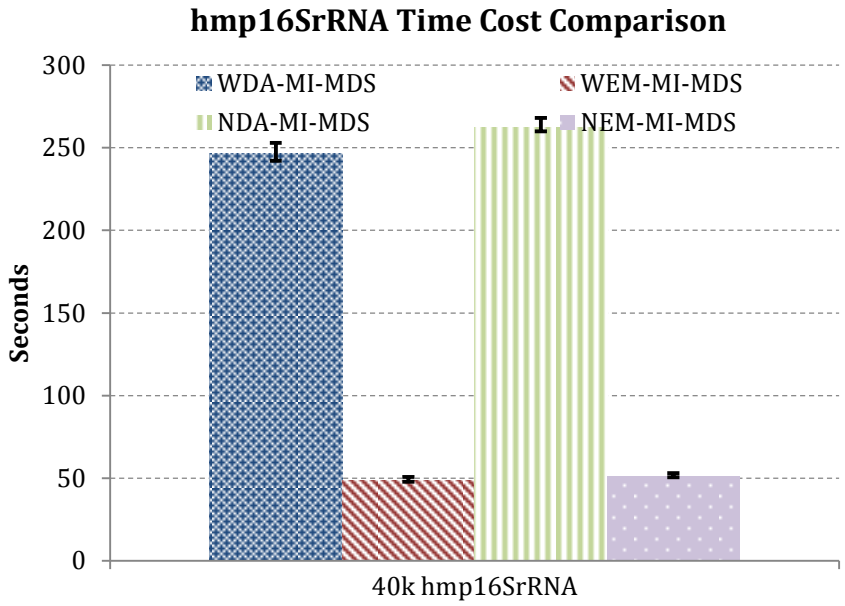

Figure 4.12: The time cost comparison between 4 MDS algorithms using 10k hmp16SrRNA sequences as in-sample data, and 40k hmp16SrRNA sequences as out-of-sample data. All 4 algorithms in this experiment are parallel algorithms using 80 cores.

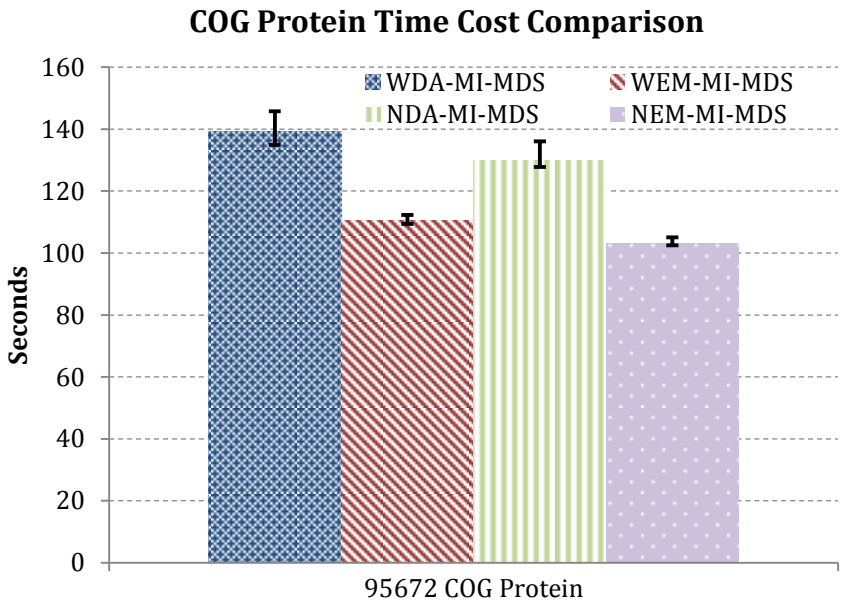

Figure 4.13: The normalized STRESS value comparison between 4 MDS algorithms using 4872 consensus sequences as in-sample data, and 95672 COG sequences as out-of-sample data. All 4 algorithms in this experiment are parallel algorithms using 40 cores.

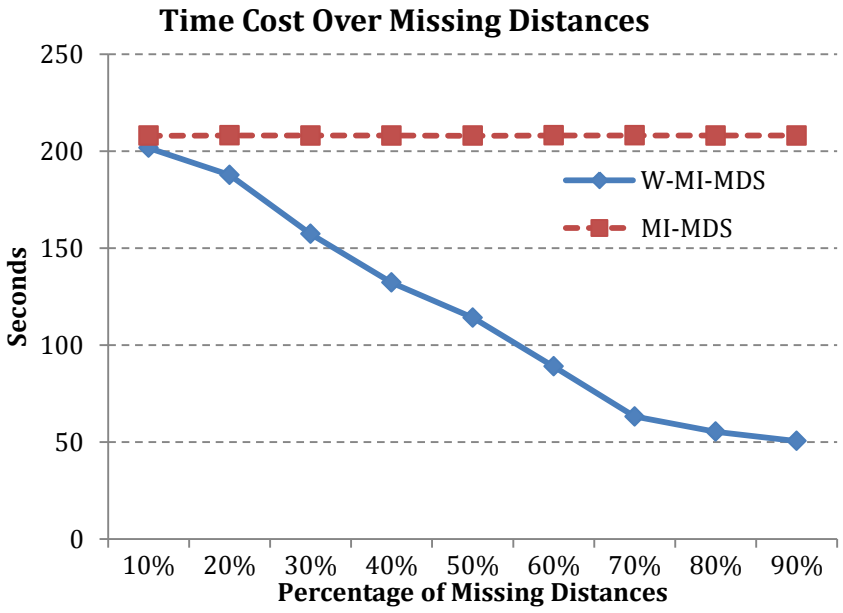

Figure 4.14: The time cost of W-MI-MDS and MI-MDS processing 40k hmp 16S rRNA data interpolating to 10k hmp 16S rRNA using 80 cores by fixing the iteration to 50 and increases the percentage of missing distances randomly.

4.5.3 **Performance of HE-MI**

The HE-MI algorithm used in this experiment includes two version, one with SSP-Tree, denoted as HE-MI (SSP) and the other is with CN-Tree, denoted as HE-MI (CN). The naïve hierarchical method is implemented using SSP-Tree, denoted as HI-MI (SSP). The experiment were tested on using 100k data selected from hmp16SrRNA data, where 10k to 50k of that data are selected as in-sample data, where the rest data are out-of-sample data and needs to be interpolated to the in-sample result. The in-sample data are processed using the DA-SMACOF algorithm (with all weights equal 1) in order to get a most accurate result. The time cost is shown in Figure 4.15. It shows that the time cost of HI-MI using SSP-Tree is much lesser than the other methods because it uses hierarchical method to avoid most of the distance computation. The time complexity of it is O(N*logN) where N is the in-sample size and N_2 is the out-of-sample size. So when the in-sample size increases, the time cost decreases because the size of out-of-sample data decreases. HE-MI (SSP) and HE-MI (CN) took around 1000 seconds to finish with 10k in-sample data while HI-MI (SSP) uses 600 seconds. When the in-sample data size increases to 50k, the time cost of HE-MI (SSP) took 1979 seconds and HE-MI (CN) took 2046 seconds. The HI-MI time cost decreases to 270 seconds. In meanwhile, the time cost of MI-MDS increases from 27000 seconds to 79000 seconds. This shows that HE-MI greatly reduced the time cost of MI-MDS, but would have a higher time cost than HI-MI method. However, the accuracy of HI-MI method is the worst among all methods compared here. This can be observed from Figure 4.16. The HI-MI always has the largest normalized STRESS value and its STRESS value remains the same from in-sample size 10k to 40k. All other methods has a decreasing STRESS value with increasing in-sample data size because of the reason mentioned above. The HE-MI (SSP) has a very similar performance with HE-MI (CN) method, and for 10k to 40k data, it has a lower STRESS value. When the in-sample data size increases to 50k, HE-MI (CN) performs better than HE-MI (SSP). The MI-MDS always has the lowest STRESS value, which is around 15% better than the other two HE-MI methods, but its time cost is around 27 to 67 times larger than them.

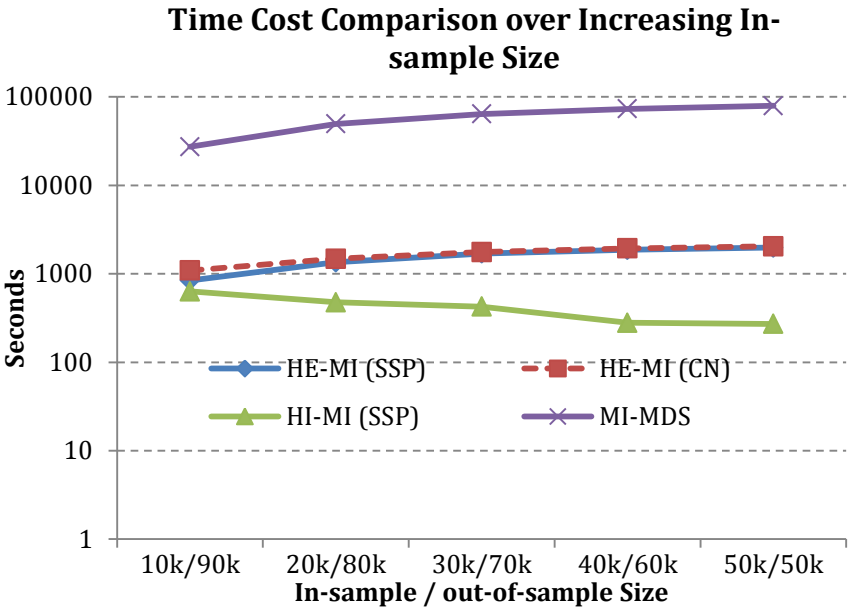

Figure 4.15: The time cost comparison of 4 MDS interpolation methods using 100k hmp16SrRNA data, and divided into in-sample and out-of-sample datasets. The in-sample dataset increases while out-of-sample decreases.

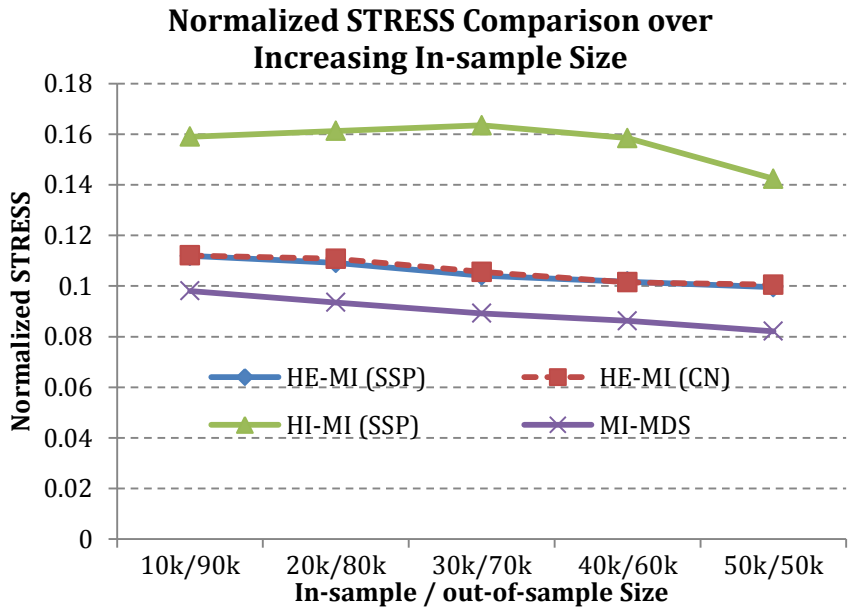

Figure 4.16: The normalized STRESS value comparison of 4 MDS interpolation methods using 100k hmp16SrRNA data, and divided into in-sample and out-of-sample datasets. The in-sample dataset increases while out-of-sample decreases.

4.6 Conclusion

This section mainly proves two optimizations to the MDS interpolation problem (also referred to as in-sample and out-of-sample problem). First optimization is adding weighting support for an algorithm similar to the majorization process from SMACOF algorithm, so called W-MI-MDS (weighted majorizing interpolative MDS). This algorithm successfully added support for distances generated from in-sample to out-of-sample with various weights, including missing distance values. From the performance analysis, it shows that if the data has different weights (instead of all 1) associate with each pair of distances, the W-MI-MDS will always yield the result with highest accuracy. Also, if there are portion of original distances generated from sequence alignment considered as missing, W-MI-MDS will reduce the time cost as well. The second optimization was focused on the hierarchical method that introduced with MDS interpolation. This problem originates from the necessity of finding k-NN in-sample points while interpolating an out-of-sample point. And for those whose distance computation takes a long time, such as sequence alignment, the hierarchical method proposed here can significantly reduce the time cost compare to the linear method by sacrificing the accuracy. Furthermore, a heuristic optimization has been proposed using tree-structure to improve the accuracy. Finally, the algorithm proposed as HE-MI has a sub-linear time cost with an approximate result with original method.

Chapter 5. DETERMINE PHYLOGENETIC TREE WITH VISUALIZED CLUSTERS

5.1 Overview

The traditional Phylogenetic Tree generation method uses multiple sequence alignment first, then followed by some traditional method, such as with RAxML. The displaying method of phylogenetic tree is limited to the several tree diagrams solely in 2D or 3D. Traditional tree display software, such as MEGA6 [76], Seaview [77] and FigTree [78] only display trees separately from the clustering result, so it is difficult to observe the relationships between the phylogenetic tree and the clustering result. Note that a phylogenetic tree automatically constructed on a single core is usually with dozens of sequences since the computation cost of the high accuracy algorithms are high, such as a famous maximum likelihood method RAxML [45]. If the number of sequence increases, the time cost of generating the tree as well as doing the multiple sequence alignment will increases dramatically. One may need takes days to process thousands of sequences automatically. Clustering, on the other hand, has a lower accuracy to classify the gene families than the phylogenetic analysis, but it cost much less time with the pairwise sequence alignment and traditional hierarchical clustering method. Once the clustering result is projected in a 2D or 3D space, one can either determine or project the phylogenetic tree using the same sequence data into the clustering mapping result.

More specifically, by using DACIDR [46], each sequence is represented as a point in the target dimension space, i.e. the 3D space. Also, by using RAxML, all the sequences are represented as leaf nodes in the phylogenetic tree. Therefore each leaf node in the phylogenetic tree corresponds to a point in the 3D dimension reduction result. Here we propose a combined method to address those limitations. For clustering, DACIDR uses a multidimensional scaling (MDS) technique to visualize sequence similarity among all sequences in a dataset as a way to infer clusters of similar sequences directly, without the need to define a sequence similarity-threshold (we will refer to this method as *MDS cluster visualization*). Because MDS cluster visualization allows the observation of sequence similarity of datasets directly, it is a promising technique for determining sequence clusters from high throughput sequencing. However, it is unclear how accurately groups of similar sequences found with the visualization correspond with defined taxonomic groups. In order to evaluate the taxonomic accuracy of groups identified with MDS cluster visualization, a phylogenetic tree was created using maximum likelihood based methods on the same sequence dataset. This tree is created by using multiple sequence alignment on the same dataset first, then use RAxML software to generate the tree in a text format. By another method referred to as Interpolative Joining, this tree can be determined in the MDS clustering in the target dimension. So the clustering and tree's evolutionary path can be viewed simultaneously. This novel approach allows the clear observation of clustering and phylogeny so that hidden structures from a tree can be exposed in the 3D space.

5.2 Related Work

As the process of determine phylogenetic tree with cluster visualization involves both clustering and phylogenetic tree displaying method, so in this section, related work of both of these two areas were discussed. Different from traditional hierarchical clustering methods, clustering methods that use inferences about phylogenetic relationships between sequences also do not require defined sequence similarity thresholds, although these methods are more computationally intensive because they require multiple sequence alignment. The Generalized Mixed Yule Coalescent (GMYC) method [79] is the most widely used of these clustering techniques. From the pattern of

single gene evolution (the coalescent) derived from a given sequence dataset (like for the 28S rRNA gene), GMYC uses a maximum likelihood approach to determine the transition point sequence changes representing speciation (Yule) events to those representing coalescent events (population divergence within the same species) [80]. A recently proposed alternative to the GMYC method is the Poisson Tree Process (PTP), which is computationally faster than the GMYC method while also achieving increased clustering accuracy [81]. The PTP estimates species clusters using a maximum-likelihood phylogenetic tree produced from the sequences as a guide (instead of the coalescent tree required for the GMYC method), and assumes that each nucleotide substitution has a fixed probability of being the basis for a speciation event. The PTP is able to give accurate species determinations regardless of the amount of sequence similarity between the species being compared. However the PTP still requires either multiple sequence alignment or a guide phylogenetic tree in order to cluster sequences, and therefore is computationally more costly than a clustering algorithm that uses pairwise sequence alignment. The methods used for phylogenetic tree creation have become more standardized compared to clustering techniques. The most commonly accepted methods are probabilistic approaches including maximum likelihood (RAxML) [45] and Bayesian [44] methods . Because both of these methods incorporate uncertainty phylogenetic tree construction, they are thought to provide phylogenies that are closely aligned with actual patterns of evolutionary history.

In terms of displaying methods for phylogenetic trees, there are many innovative displaying methods proposed during the past few years [82]. Mega6 [76], Seaview [77] and FigTree [78] are some popular software that enables visualizing and editing trees in 2D. These type of software provides a full solution of manipulating phylogenetic trees so that it can be displayed in cladogram, phylogram or even more tree diagrams. There are some existing software that enables displaying phylogenetic tree in 3D as well. Paloverde [83] is a program designed to visualize the phylogenetic structure in an interactive virtual 3D world. It implements radial 2D layouts and spiral 3D layouts. It can display up to 2500 leaf nodes of a given phylogenetic tree. H3 [84] is software that uses hyperbolic structure to visualize phylogenetic tree in 3D so that more nodes could be displayed simultaneously. It successfully displayed over 20,000 nodes by using

hyperbolic navigation to reduce visual clutter. Arena3D [85] introduces a new concept of staggered layers in 3D space. It combined multiple clustering and tree construction algorithms in order to handle large scale phylogenetic tree with multiple connections. Its novel approach can successfully identify some hidden information within a phylogenetic tree that could not be found within 2D structure. TreeTracker [86] proposed a general strategy to determine the congruence between a hierarchical and a non-hierarchical classification. It finds the sequences clusters based on a constructed phylogenetic tree in 2D, usually as a circular cladogram. It avoids overrepresentation by considering the likelihood of the topology of phylogenetic tree.

5.3 Phylogenetic Tree Visualized in 3D

As mentioned previously, by using DACIDR, each sequence is represented as a point in the target dimension space, i.e. the 3D space. Also, by using RAxML, all the sequences are represented as leaf nodes in the phylogenetic tree. Therefore each leaf node in the phylogenetic tree corresponds to a point in the 3D dimension reduction result. To display the clustering result with phylogenetic trees, one way to do that is to display the clusters of sequences on one side, then the phylogenetic tree on the other side. This can be generated using a cladogram, where each point from the tree node is projected into a corresponding point in the clustering result. The other way is to display these two result tightly coupled, so that each point in clustering result directly represents a sequence from the tree. These two methods are discussed separately in the following section.

5.3.1 Cuboid Cladogram Generation

As the clustering result is generated in a 3D space, one intuitive way to view the clustering result and the tree together is to display the clustering result on one side, while the tree is displayed on the other side. This method is solely used to verify the result between the clustering and phylogenetic analysis generated in a separate method using the same sequence set. The cladogram from a purely tree diagram can be project to the clustering result so that each point in the target dimension space correspond to a leaf node in the phylogenetic tree. The clustering result is also referred to as the MDS dimension reduction result (MDS Clustering). Instead of building tree from top down (like what most algorithm would do), the tree can be built from the MDS clustering

result, i.e. from down to top. Once the phylogenetic tree is generated from a traditional method, the tree can be projected onto the clustering result.

To find the suitable plane for the phylogenetic tree to be projected to, one can either randomly select a plane or use the best possible plane. The random selection may has a problem that the tree is projected to a plane that points are not clearly separated. Figure 5.1 shows an example of a phylogenetic tree with 8 sequences and the same 8 sequences with clustering result. Assume that this tree is constructed using a traditional method, and if one needs to display the tree simultaneously with the clustering, one needs to choose which dimension it wants to be projected to. In current example, as the both the clustering and the trees are in 2D, one needs to project the tree into a one dimension plane (a line) within the 2D clustering where the points are lied on. A random choice could end up choosing a tree shown in Figure 5.2. It shows that the tree is not well projected, and the lines were overlapping with each other so that it hard to observe the connections between the leaf points.

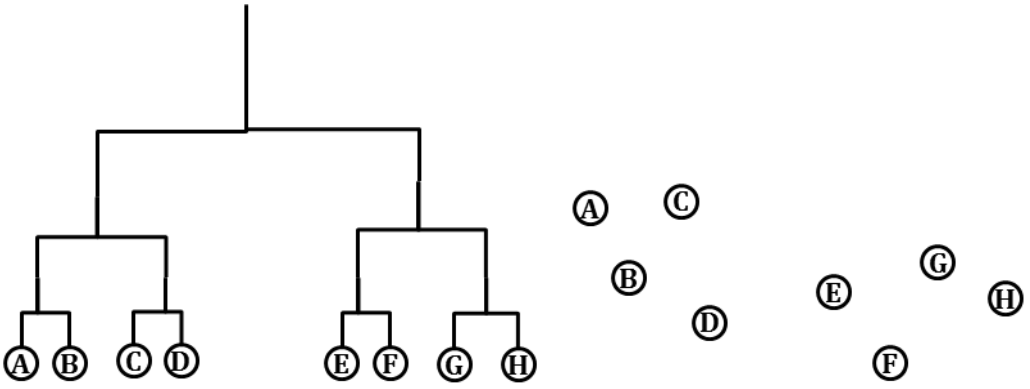

Figure 5.1: The left hand side of the graph representation is a cubic cladogram displayed with 8 sequences. The right hand side of the graph is the same 8 sequences visualized in 2D after dimension reduction.

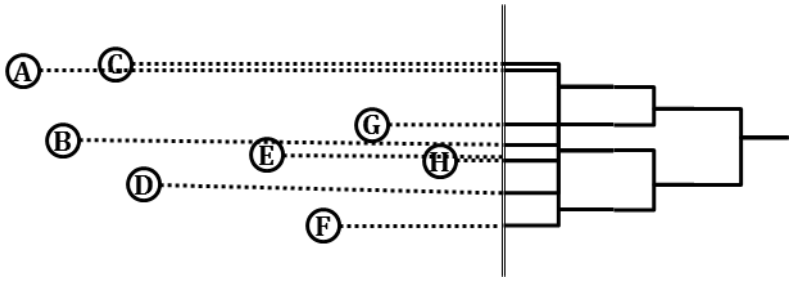

Figure 5.2: The example of choosing a random line to project all the sequences to and draw the given cubic cladogram accordingly.

And in practice, an obviously good choice of the line would be the same line as shown in Figure 5.3. This is because the points that projected on this line are spread out instead of crowded together as shown in Figure 5.2. One way to select a best possible plane is to use the principal component analysis (PCA) [87] on that. The PCA is a popular method that used for dimensionality reduction based on the vectors in the original dimensionality, which is 2 in this example. It can transform the data to a new coordinate system such that the greatest variance comes to a certain coordinate (usually the first one). Suppose the target dimension for the clustering result is in L-dimension and there are number of N points, the MDS clustering coordinates can be represented in an N×L matrix, denoted as X. The general solution for PCA involves singular vector decomposition on the original matrix X. In in the example given as in Figure 5.1, x is a 8×2 matrix. So there are 2 dimensions generated by PCA that gives the largest variance and the smallest variance of the coordinates on that dimension. The Figure 5.4 actually shows the line which gives the largest variances. Compared to the random choice shown in Figure 5.2, the projected tree is much clearer, and the correlations between the clustering result and the phylogenetic tree can be intuitively observed.

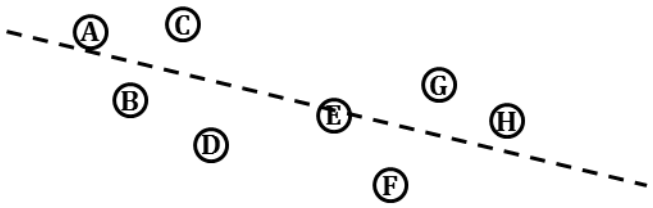

Figure 5.3: An example of a good choice of projection line as the dotted line within 8 sequences visualized in 2D space.

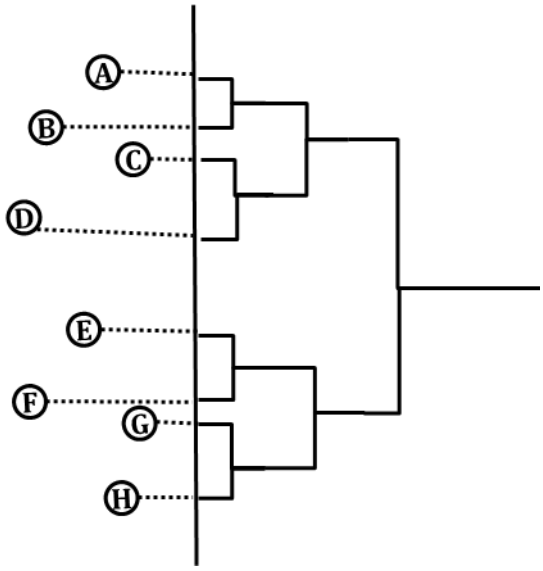

Figure 5.4: The example of choosing a good projection line determined by PCA to project all the sequences to and draw the given cubic cladogram accordingly.

As shown in the 2D plot, the branches sometimes can overlap each other if a projection from clustering to a certain line is made. This could cause in-efficient display if some dimension similar to Figure 5.2 is chosen. Additionally, one cannot guarantee the clean coordinates system such as in Figure 5.4 exists in every clustering result and phylogenetic tree result. In this particular example, all the sequences has the same order as from clustering result to phylogenetic tree. If they have different order, the branches from the projected tree may be overlapping with each other, e.g. if sequence A and sequence E swap location. So to avoid that, one more dimension can be added to this method. If the clustering result is shown in 3D and the tree is projected onto a 2D plane, the overlapping of the branches won't be an issue since the branches are in 1D. The example graph is shown in Figure 5.7. This graph is referred to as cuboid cladogram. This is because the phylogenetic tree displayed in the graph won't show difference lengths between the leaf nodes and their parents. The phylogenetic tree here could be a rooted tree and an outgroup can be added as well. Once the plane with the largest variance is found for the 3D coordinates, the points are projected onto that plane. Then each pair of points, which correspond to each pair of leaf nodes shared the same parent are selected in the plane, their parent will be a new point which is the exact middle point of the connection between these two points. The parent will be draw one level higher

than its children. The process is recursively done until all points, including the root and the outgroup points are drawn.

Figure 5.7, Figure 5.8, and Figure 5.9 illustrates the example of cuboid cladogram in 3d with the MDS clustering result. This result is generated from 446k fungal data, where 126 representative sequences from each cluster is selected, along with 74 sequence from GenBank shown in different color. The screen shot are from two different angel, one is from the side of the tree and the other is from the top of the tree. From the these figures, the correlations between the phylogenetic tree and the clustering can be easily observed. And in this graph, it shows that the phylogenetic tree and the MDS clustering have very high correlation.

To summarize, the cuboid cladogram took the following steps to generate: 1) generate the phylogenetic tree using traditional method; 2) generate the MDS clustering result; 3) Use PCA to find the plane with coordinates that has largest variance; 4) project the points from clustering onto the plane; 5) generate the cladogram from the points on the plane. Finally, by viewing the tree plot in 3D along with the clustering result, the clustering and phylogenetic analysis can be done simultaneously.

5.3.2 **Spherical Phylogram Generation**

The Cuboid Cladogram is one way to display the clustering and the tree together by displaying the clustering result on one side and project all tree leaf nodes to the clustering result by selecting the plane with largest variance of all coordinates. But in some cases, the points may overlap with each other once a projection is done on a selected plane. It will be confusing if clear observation is needed between the phylogenetic analyses and clustering analysis. Moreover, the cladogram does not preserve the branch lengths between each pair of internal nodes. Therefore, in this section, a new plot called Spherical Phylogram [63] (SP) is proposed. The method displays an existing phylogenetic tree by using the clustered sequences from the same dataset directly as leaf nodes of the tree. This allows for direct visual comparison between the phylogenetic tree and the sequence clusters. The generated tree can is shown in 3D, and the branch lengths correspond to the dissimilarities between each pair of the internal nodes.

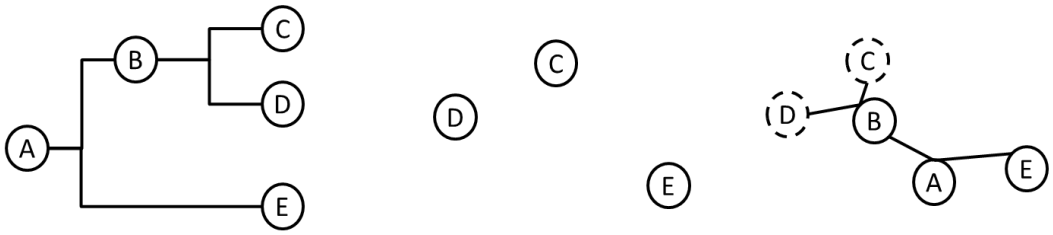

Figure 5.5: The example of distance calculation in a phylogenetic tree with 3 leaf nodes and 2 internal nodes.

The internal nodes cannot be directly observed because they represent hypothetical ancestor sequences, and therefore the distances from internal nodes to leaf nodes of the generated phylogenetic tree are unknown. By using RAxML, it is possible to calculate distance from an internal node to another node by using the summation over all the branches between them. For example, in Figure 5.5, the distance between point C and E can be calculated by summing over branch(C, B), branch(B, A) and branch(A, E). This distance calculation can generate a pairwise distance matrix for all the nodes based on all the branch lengths. However, the sum of branch lengths does not work to find the distance between pairs of leaf nodes since the pairwise distances between leaf nodes are already known from the MDS cluster visualization results. For example, the distance between leaf node C and D shown in Figure 5.5 is clearly not equal to branch(B, C) + branch(B, D). Therefore if the summation over the branches is used for defining distances during interpolation, the result will have a high bias because different distances were used for leaf nodes. Therefore, we chose the distance calculation method used in neighbor joining [42] (NJ) algorithm to calculate the distances between internal nodes based on the existing distances between leaf nodes so that all distances used for visualization are consistent, which is also referred to as tree distance.

The NJ algorithm starts with a completely unresolved tree, whose topology corresponds to that of a star network, and ends once the tree is completely resolved and all branch lengths are known. The core idea of this algorithm is to find a way of constructing a tree that follows the balanced minimum evolution (BME) criterion, which generates the optimal tree topology and minimizes the branch lengths of the tree. Therefore we use the same strategy to interpolate the phylogenetic tree

into the MDS cluster visualization result to generate a SP that will have a minimum total branch length. Nevertheless, if the SP matches the original phylogenetic tree better, the sum of all the branches will be shorter.

The distance calculation used here is similar to the one used NJ, and it can be formulated according to the following: suppose we have n existing points, denoted as $P = \{p_1, p_2, p_3, \dots, p_n\}$. And a point p_i can be represented as a vector $[x_{i1}, x_{i2}, \dots, x_{iL}]$ in L-dimensions. The distance between two points p_i and p_j is denoted as $d(p_i, p_j)$ and can be calculated as Euclidean distance using the following equation:

$$d(p_i, p_j) = \sqrt{\sum_{l=1}^{L}(x_{il} - x_{jl})^2} \quad (59)$$

Given any two points $p_i, p_j \in P$, there are two corresponding leaf nodes in the phylogenetic tree. Their parent is denoted as a new point $\hat{p}$ that can be interpolated into the target dimension space. The distance from $\hat{p}$ to p_i and p_j can be given in the following equations:

$$d(\hat{p}, p_i) = \frac{1}{2}d(p_i, p_j) + \frac{1}{n-2}\sum_{k=1}^{n}(d(p_i, p_k) - d(p_j, p_k)) \quad (60)$$

Because all of the distances follow three basic rules for Δ mentioned in Section 1.2, all distances are symmetric, i.e. $d(p_i, p_j) = d(p_j, p_i)$, and $d(\hat{p}, p_i)$ can be calculated as

$$d(\hat{p}, p_j) = d(p_i, p_j) - d(\hat{p}, p_i) \quad (61)$$

The distances from p_to all other points, except p_i and p_j, can be obtained using the following equation where $1 \leq k \leq n$ where $k \neq i$ and $k \neq j$:

$$d(\hat{p}, p_k) = \frac{1}{2}(d(p_i, p_k) + d(p_j, p_k) - d(p_i, p_j)) \quad (62)$$

Note that equation (59) is the Euclidean distance calculation and equation (60) to equation (62) are the calculation of the minimum evolution path for any given two points in P, so that for any internal node in the phylogenetic tree, its distance to all other points can be obtained using the equations above.

Algorithm 5 Interpolative Joining algorithm

Input: $\boldsymbol{P}, \overline{\boldsymbol{P}}, \boldsymbol{T}, \overline{\boldsymbol{T}}$
Output: P as the spherical phylogram

1. For each pair of siblings ($\boldsymbol{t_i}$, $\boldsymbol{t_j}$) in T
2. Find their parent $\hat{\boldsymbol{t}}$ in $\overline{\boldsymbol{T}}$
3. Find point $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ in P
4. For other point $\boldsymbol{p_k}$ in P
5. Compute $\boldsymbol{d}(\boldsymbol{p_k}, \boldsymbol{p_i})$, $\boldsymbol{d}(\boldsymbol{p_k}, \boldsymbol{p_j})$ using (4)
6. End for
7. Compute $\boldsymbol{d}(\hat{\boldsymbol{p}}, \boldsymbol{p_i})$and $\boldsymbol{d}(\hat{\boldsymbol{p}}, \boldsymbol{p_j})$ using (5) and (6)
8. For other point $\boldsymbol{p_k}$ in P
9. Compute $\boldsymbol{d}(\hat{\boldsymbol{p}}, \boldsymbol{p_k})$ using (7)
10. End for
11. Use (8) as object function and W-MI-MDS to compute $\hat{\boldsymbol{p}}$
12. Remove $\boldsymbol{t_i}$ and $\boldsymbol{t_j}$ from T
13. Add $\hat{\boldsymbol{t}}$ into T and remove $\hat{\boldsymbol{t}}$ from $\overline{\boldsymbol{T}}$
14. Add $\hat{\boldsymbol{p}}$into P and remove $\hat{\boldsymbol{p}}$ from $\overline{\boldsymbol{P}}$,
15. End for
16. Return P

When the distances from the internal nodes to all other points are obtained, we can then interpolate the internal node as a point into the target dimension space. In our case, the points in the 3D space that correspond to the phylogenetic tree's leaf nodes are the in-sample data, denoted as P, and the points representing internal nodes are the out-of-sample data, denoted as $\overline{\mathrm{P}}$. By using equations (59) to (62), the distance of an out-of-sample point $\hat{\mathrm{p}}$ to all other in-sample points is calculated as the original distance for interpolation, which is denoted as $\hat{\Delta}$. After $\hat{\mathrm{p}}$ is interpolated to L-dimension, it can be represented as a vector $\hat{\mathrm{x}}$ with length L. Nevertheless, the in-sample points and out-of-sample points in the L-dimension can be defined as $\mathrm{X} = \{\mathrm{X}_1, \mathrm{X}_2\}$, where $\mathrm{X}_1 = \{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \dots, \mathrm{x}_\mathrm{N}\}$ and $\mathrm{X}_2 = \{\mathrm{x}_{\mathrm{N}+1}\}$. The distance from $\hat{\mathrm{p}}$ to all other points can be obtained using equation (1), which is the Euclidean distance in 3D space, denoted as d(X). So for each out-of-sample point $\hat{\mathrm{p}}$, there is a difference between the Euclidean distance in the L-dimension and the original distance, and the object function is given as in equation (3).

Equation (60) and equation (62) give the distance calculation formulas for the internal nodes, which are also referred to as the out-of-sample points in previous section, and equation (3) gives the STRESS value of using interpolation for the internal nodes. For each internal node, W-MI-MDS can be applied to find its location in the target dimension space. However, not all internal nodes from the phylogenetic tree were selected only based on the leaf nodes. Since in traditional

out-of-sample problems, the in-sample dataset remains the same during interpolation, it is not applicable to use those kinds of algorithms for internal node interpolation. The most right diagram in Figure 5.5 gives an example of how the internal nodes are interpolated during neighbor joining. Node A is interpolated based on node E and node B, which is also an internal node for the entire phylogenetic tree shown in the most left diagram.

To solve that problem, we proposed an algorithm called Interpolative Joining (IJ). In IJ, the in-sample dataset needs to be modified during the interpolation process. Because the out-of-sample points are interpolated one by one, each out-of-sample point that is already interpolated is added into the in-sample dataset and will be considered as an in-sample point for subsequent out-of-sample points. To do this, the IJ algorithm searches the tree from the bottom up. Every time two leaf nodes are found that share the same parent, those two leaf nodes are used to calculate the coordinates for the internal node. The two leaf nodes will then be removed from the tree, and the newly interpolated internal node will be considered a new leaf node. This is demonstrated in Figure 5.5, point C and D are discarded from the leaf node set once node B is interpolated. However, these two in-sample points, which correspond to the two leaf nodes, will remain in the in-sample dataset. Therefore, the total number of nodes for the input phylogenetic tree will be decreasing and the size of the in-sample dataset will be increasing during the interpolation process.

In formal definition, P and $\overline{P}$ are used in terms of in-sample and out-of-sample points in L-dimension; T is the set of leaf nodes and $\overline{T}$ is the set of internal nodes from the phylogenetic tree. Therefore p_i is the representation of t_i in the target dimension space. For each pair of leaf nodes t_i and t_j that have the same parent $\hat{t}$, there is a pair of in-sample points which are denoted as point p_i and p_j in P that represents them. Immediately after $\hat{t}$ is found, the $\hat{p}$ that represents it is initialized as a random point and added into $\overline{P}$. After $\hat{t}$ is interpolated into the L-dimension space, $\hat{p}$ is removed from $\overline{P}$ and added into P. The t_i and t_j will be removed from T, and $\hat{t}$ is added into T and removed from $\overline{T}$. Nevertheless, $\overline{P}$ will always contain only one out-of-sample point during each iteration, where the iteration number equals the number of internal nodes in $\overline{T}$ at the beginning. The detailed process of IJ is illustrated in Algorithm 5. As the calculation of Euclidean distance

and WDA-MI-MDS is very fast, generating the SP with a predefined phylogenetic tree and MDS cluster visualization result only takes a few seconds on a single core.

5.4 Performance Analysis

The experiments were carried out on BigRed II, which is a hybrid cluster with a total of 344 CPU nodes with 32 cores per node, and Quarry with a total of 2644 cores with 8 cores per node at Indiana University to process the data with the help of Twister and Hadoop. The clustering and visualization of the sequence datasets were completed using DACIDR. We created a maximum likelihood unrooted phylogenetic tree from the multiple sequence alignment (MSA) with RAxML using 100 iterations with the general time reversible (GTR) nucleotide substitution model and with gamma rate heterogeneity (GTRGAMMA). We then used the tree to guide the generation a pairwise distance matrix between all sequences in each of the two MSA datasets using RAxML. These pairwise distance matrices were then used as the reference when testing for the effect of alignment technique and sequence length on consistency between the clustering and the phylogeny. The tree is in New Hampshire format (newick) [88]. Finally, the IJ was run on a local machine to generate a spherical phylogram (SP), which can be displayed using a data visualization software called PlotViz3 [89].

We first downloaded the sequence alignment of AM fungal sequences from a recent large-scale phylogeny of AM fungi [47] and only retained sequences that contained at least a portion of the 28S rRNA gene. We then collected two sets of additional AM fungal sequences: (1) sequences from GenBank [48] that had confident species attribution in order to supplement the species coverage within the sequence dataset; (2) representative sequences for known AM fungal species obtained from spores using 454 sequencing of the variable and phylogenetically informative D2 domain of the 28S rRNA gene. We applied DACDIR on this dataset to find 126 clusters and then picked a representative sequence for each cluster as part of the dataset. The additional sequences from GenBank were added to the original sequence alignment from [47] using MAFFT [90]. In order to evaluate how different sequence lengths affected the correspondence between phylogenetic trees and clustering, we then created two datasets with sequences that shared the

same starting location on the 28S rRNA gene: one dataset contained longer sequences referred as 999nts, and the other contained shorter sequences, referred as 599nts. The 999nts contains 801 sequences from [47] and 505 sequences from GenBank for a total of 1306 sequences, and 599nts contains 514 sequences from [47], 380 sequences from GenBank, and 126 representative 454 sequences for a total of 1020 sequences. The detail of how these sequences were aligned and chosen can be found in [63]. The RAxML took about 4 hours to finish on the first dataset and 7 hours to finish on the second dataset using 8 cores. And the MDS only took a few minutes to finish on the same dataset using same amount of cores. The tree generated from MSA and RAxML is illustrated in Figure 5.10 using FigTree [78]. The MDS cluster visualization is shown in Figure 5.6. The gene family listed shown that this tree is generated with high precision, and none of the sequences are miss-classified. The cuboid cladogram generated using the same dataset is given in Figure 5.7, Figure 5.8, and Figure 5.9 from 3 different angles as an example. Figure 5.11 and Figure 5.12 are two examples of the screenshots over a spherical phylogram generated using the same dataset and MDS method using PWA and MSA.

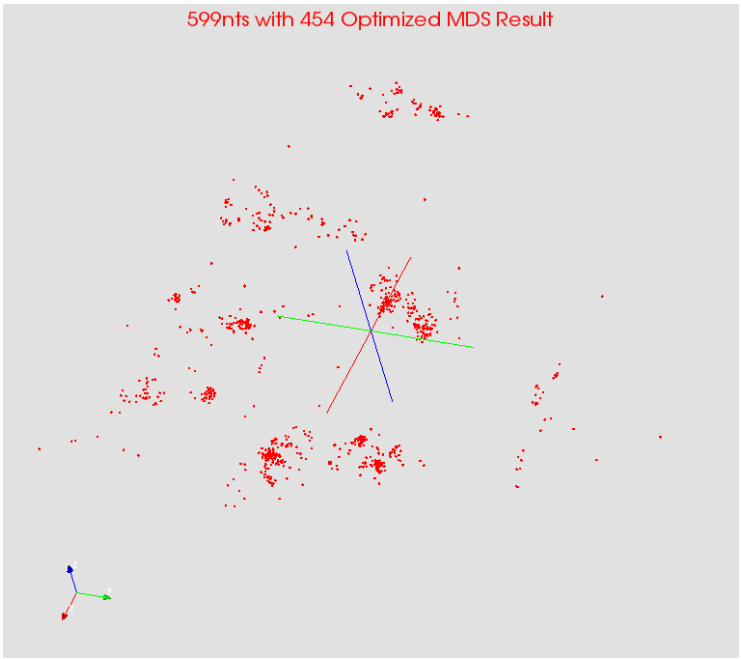

Figure 5.6: The visualization result of 599nts data using MSA and WDA-SMACOF

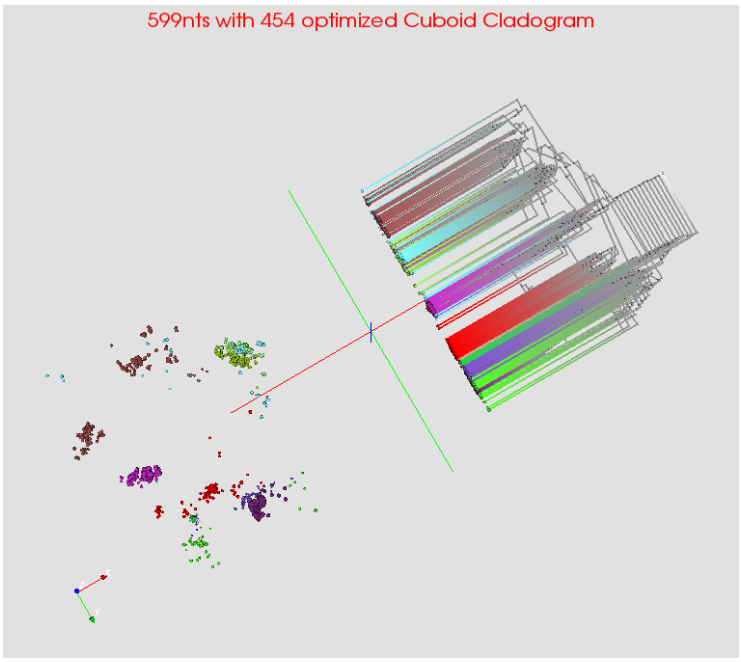

Figure 5.7: The screen shot from the side of the cuboid cladogram by choosing a plane using PCA on 599nts data using MSA and WDA-SMACOF

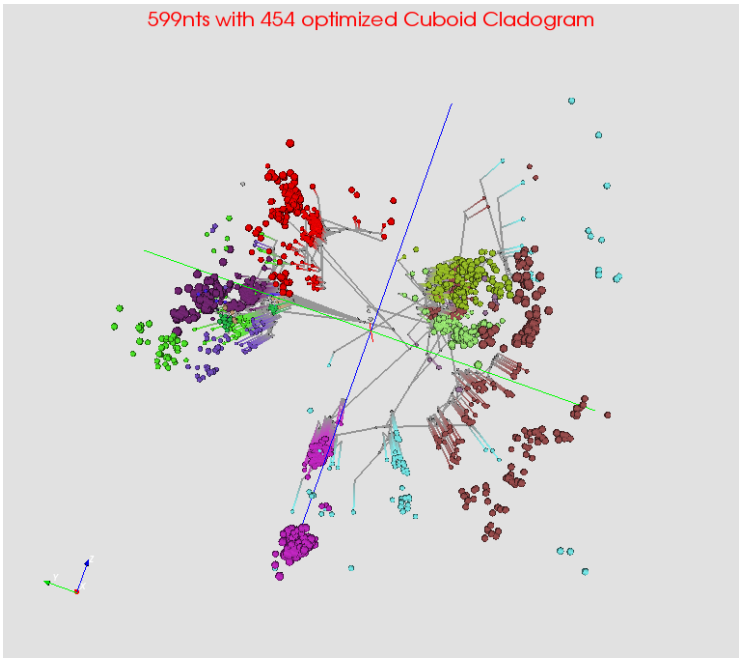

Figure 5.8: The screen shot from the bottom of the cuboid cladogram by choosing a plane using PCA on 599nts data using MSA and WDA-SMACOF

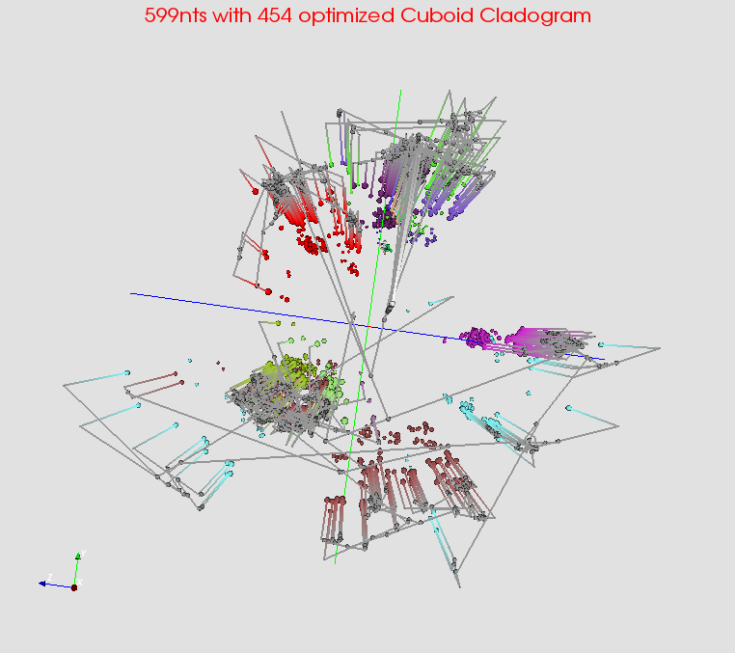

Figure 5.9: The screen shot from the top of the cuboid cladogram by choosing a plane using PCA on 599nts data using MSA and WDA-SMACOF

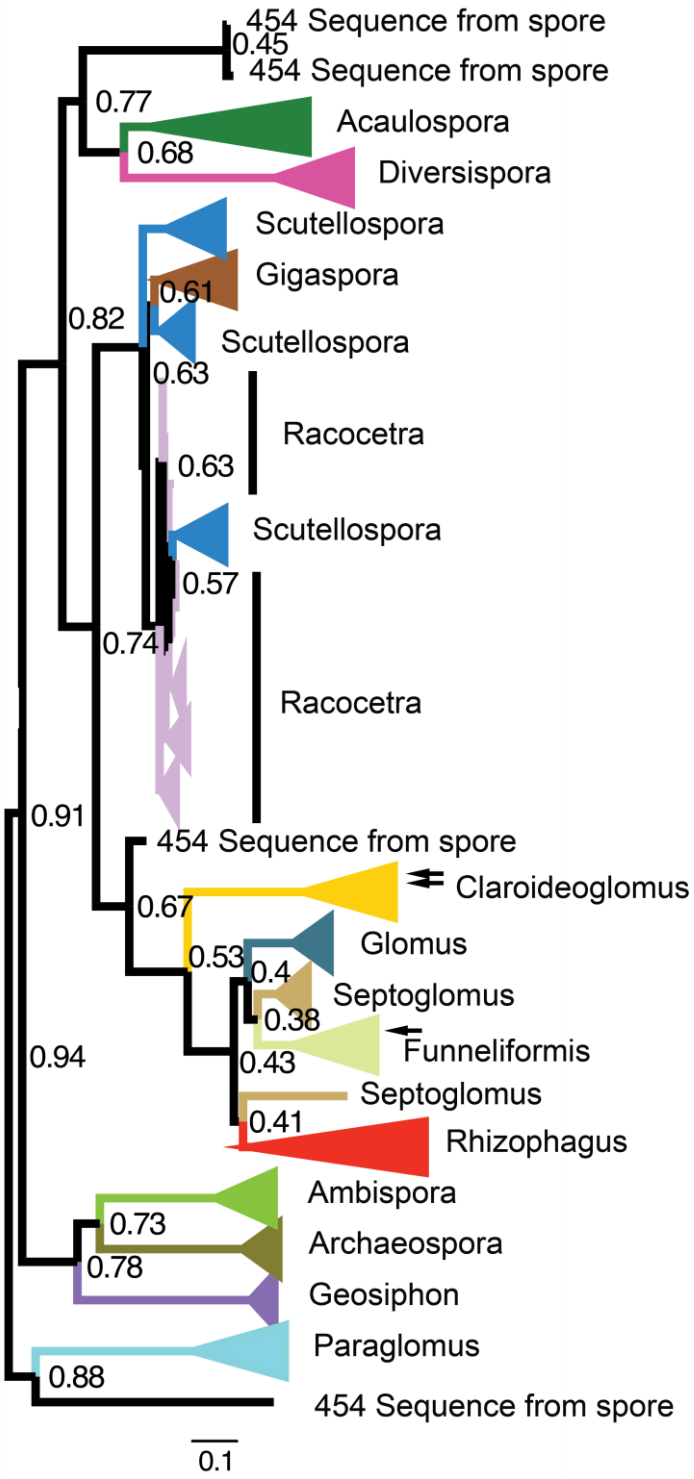

Figure 5.10: Maximum likelihood phylogenetic tree from 599nts that is collapsed into clades at the genus level as denoted by colored triangles at the end of the branches. Branch lengths denote levels of sequence divergence between genera and nodes are labeled with bootstrap confidence values. 454 sequences from spores that are not part of another clade are denoted with the label '454 sequence from spore'. Distance Calculation Comparison

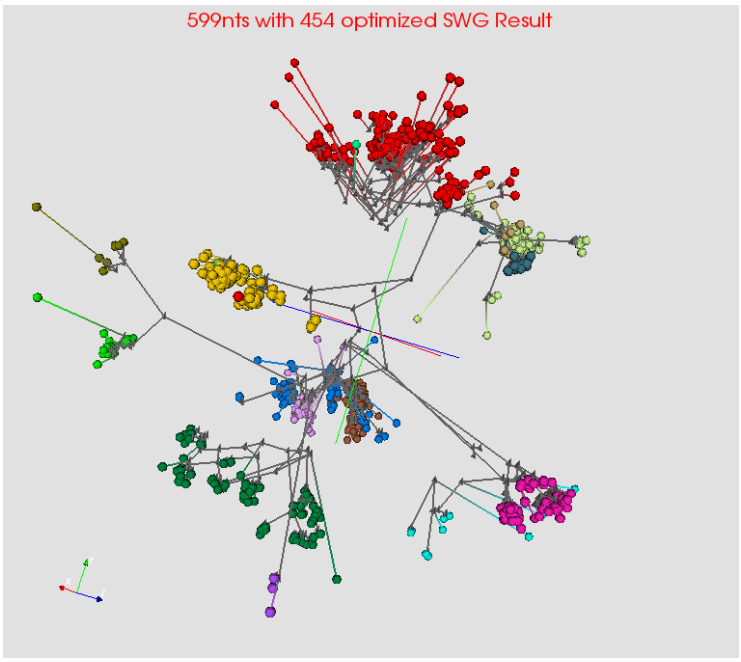

Figure 5.11: The screenshots of spherical phylogram for using the phylogenetic tree shown in Figure 5.10 SWG pairwise sequence alignment. The colors of the branches in these figures are as same as the colors of the branches shown in Figure 5.10.

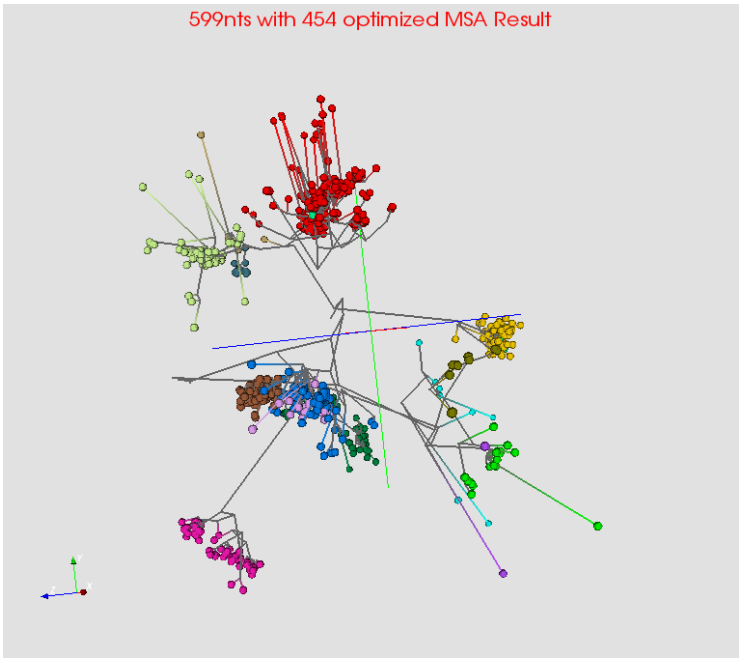

Figure 5.12: The screenshots of spherical phylogram for using the phylogenetic tree shown in Figure 5.10 multiple sequence alignment. The colors of the branches in these figures are as same as the colors of the branches shown in Figure 5.10.

5.4.1 **Distance Calculation**

The distance calculation from different sequence alignment usually yields different result for dimensionality reduction. A general impression from the phylogenetic community is that MSA performs better than PWA when doing the phylogenetic analysis. And PWA alignment is usually used for sequence clustering while data size is relatively large. Two popular choice were compared, Needleman Wunsch (NW) and Smith-waterman Goth (SWG). The comparison result is listed in Figure 5.13. The performance evaluation used Mantel test to determine whether pairs of experimental treatments retained the same structure of sequence differences between them. Mantel tests determine whether a correlation between the entries contained in two different pairwise distance matrices is statistically significant by permuting the distance matrices to obtain an empirical p-value for the correlation. The treatments consisted of different alignment techniques applied to each of the two different length datasets; comparisons were then made to the RAxML distance matrix from the same dataset. The Mantel tests were performed using the vegan package in R (version 3.0.2, R Core Team 2013), and none of the tests had p-values greater than 0.001, suggesting all of the measured correlations were likely significant despite the increased type I error (false-positive) rate that can occur with Mantel tests [91].

Figure 5.13 illustrates the result of the Mantel test applied on MSA and PWA which includes both the SWG and NW. Using longer sequences (999nts) consistently resulted in higher correlations between the reference distance matrix and either of the pairwise alignment techniques. However, both the SWG and the NW pairwise alignment methods gave comparable correlation values for 599nts and for shorter sequences (599nts). The very high correlations between the RAxML reference matrix and the MSA distance matrix used for MDS cluster visualization regardless of sequence length are expected because the input alignment is identical for both matrices and only the distance calculation method is different. Using pairwise alignments for the same datasets resulted in lower correlations with the RAxML reference matrix, although they still provided a reasonably good fit.

The relationships between genera of AM fungi from the phylogenetic tree created with 599nts in Figure 5.10 was consistent with the current understanding of AM fungal phylogenetic

relationships, with the exception of Racocetra, Scutellospora, and Gigaspora all being assigned to the same evolutionary group. By comparing the phylogenetic tree in Figure 5.10 and the SPs in Figure 5.11and Figure 5.12, it is possible to visualize the how the branches of the tree correlate with the sequences after MDS. If long branches are required in the interpolated tree in order to connect points that are the same color in the MDS visualization, then the tree does not match well with the MDS result. This is because the sequences on the same branch of the phylogenetic tree are more similar to each other than to other sequences in the dataset, and therefore they should be located close to each other in the MDS visualization as well. The SPs show that the points with the same color (as color-coded from the genera in the phylogenetic tree), generally group together. There are a few points in the SPs using pairwise alignments (SWG) that have longer branches than the points from the SP using the MSA. This is consistent with the fact that the SP generated from the MSA has a better correlation with the phylogenetic tree than the SPs generated from the pairwise alignments. However, the SPs also verify that using pairwise alignments for the MDS generally gives a good fit with the interpolated phylogenetic tree.

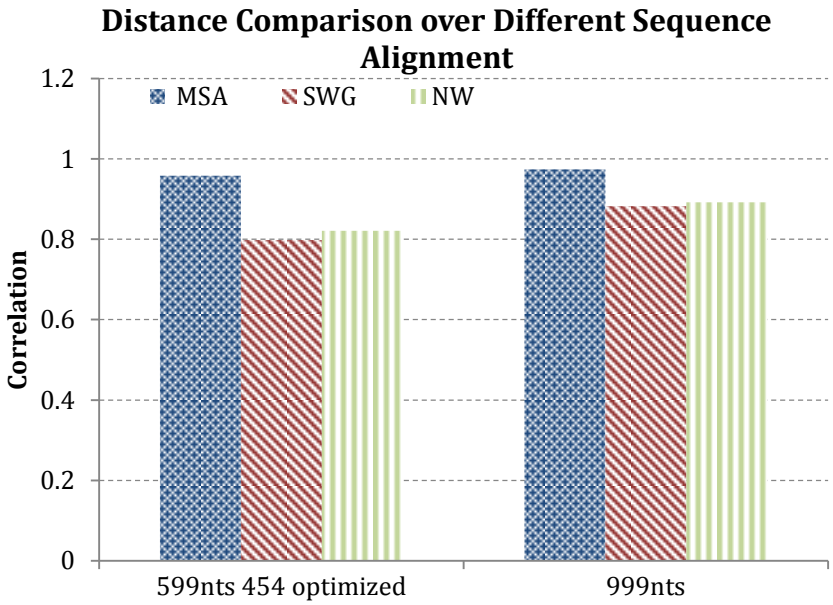

Figure 5.13: The comparison using Mantel between distances generated by MSA and two PWA methods and RAxML

5.4.2 **Dimension Reduction Methods Comparison**

The different methods of MDS affected how well the phylogenetic tree projected using IJ matched the sequences in 3D. WDA-SMACOF is a robust MDS method that can reliably find the global optima, whereas EM-SMACOF can be easily trapped under local optima. The LMA usually had a result that was very similar to EM-SMACOF as shown in Figure 5.14 and Figure 5.15. The normalized STRESS value for each different input using the different methods was from 0.021 to 0.023, which suggests the distances after dimension reduction have a high similarity to the original distances, and therefore sequence differences were preserved well during MDS; WDA-SMACOF always had the lowest STRESS value compared to the other two methods.

Figure 5.14 and Figure 5.15 illustrates the result from Mantel Test on 599nts data and 999nts data. From this figure, it shows that the difference over these three methods does not vary much. But WDA-SMACOF always shows a better correlation on all dataset using all different alignment methods. Averagely, WDA-SMACOF shows a 1.4% better result than LMA and 1.7% better than EM-SMACOF on 599nts data. For 999nts data, WDA-SMACOF has a higher correlation with RAxML than the other two methods by 5.4%.

In order to show the result with larger divergence, the summation over all the branch lengths of the phylogenetic tree in the SP is used here to evaluate the differences between these three dimension reduction methods. This is a much more sensitive method than the Mantel Test as shown in Figure 5.16 and Figure 5.17. As mentioned before, the points of the dimensional reduction that connect to the same branches of the SP should be shorter if they match the tree better, which will result in a lower sum of branch lengths. WDA-SMACOF had a much lower sum of branch lengths compared to both LMA and EM-SMACOF. This is because the clusters naturally appeared when the STRESS value became lower, but LMA and EM-SMACOF were trapped under the local optima, so there are some points from very small branches of the tree could still be far away from each other in the 3D space and not clustered. In contrast, WDA-SMACOF can reliably find the global optima so that these points from very small branches are always converged into clusters. This is why there were not any excessively long branches for the SP plots generated by using WDA-SMACOF as shown in Figure 5.11 and Figure 5.12. WDA-SMACOF has a result better than LMA

on an average basis of 15.6% and is 8% better than EM-SMACOF on 599nts data. For 999nts data, this difference is even larger, where the WDA-SMACOF is 30% better than the other two methods.

From above experiment results, it is safe to conclude that the sum over branch lengths is a more sensitive measurement than the Mantel test while evaluating the SPs. However, it also has a higher variance than the Mantel test because it was calculated after IJ. On the other hand, the Mantel test is more robust and shows very little differences while comparing the dimension reduction methods. Therefore, we use both the sum of branch lengths and pairwise correlations from the Mantel test to demonstrate that the interpolated phylogenetic trees closely fit the MDS using WDA-SMACOF, even with pairwise alignments.

More detailed analysis on how the sum of branches is being affected by the quality of the MDS mapping result is shown as below. The sequence alignment is selected as MSA, two sequences, with the ID number of FR750020_Arc_Sch_K and FR750022_Arc_Sch_K. these are two sequences from a same family, which should be close to each other. And in a perfect dimension reduction result, these two sequences should form a small cluster, and away of other sequences. The distances from these two sequence to all other sequences in the same dataset confirms that. This is a dataset includes 831 sequences from 599nts data, excluding th esequences from 454 optimized representative sequences. This figure shows a tree generated based on the same RAxML result as shown in Figure 5.18, and use the LMA dimension reduction method. Although sequence FR750020_Arc_sch_K and FR750022_Arc_Sch_K belongs to a same family, they are still far away from each other because the cluster conntains only 2 sequences. And for a dimension reduction method that trapps under local optima, it will converge when the large clusters are projected corretly, without projecting these small clusters into correct locations. But for WDA-SMACOF, this will be avoid since WDA-SMACOF always find the global optima. Figure 5.19 shows the same tree of the same dataset generated using WDA-SMACOF. And the highlighted sequences are now near each other in the MDS clustering result. Since the branch will be very long if these type of sequences (some sequences in blue square also suffers the same problem) are

far away from each other in the 3D mapping, so a better mapping has lower sum of branch lengths, as shown in Figure 5.18.

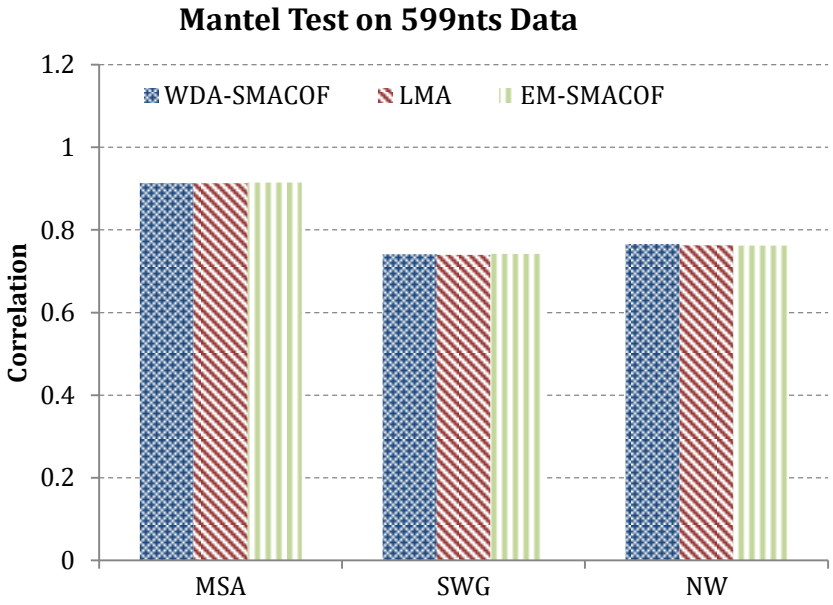

Figure 5.14: Mantel comparison of WDA-SMACOF, LMA and EM-SMACOF using distance input generated from one MSA method and two PWA methods on 599nts dataset

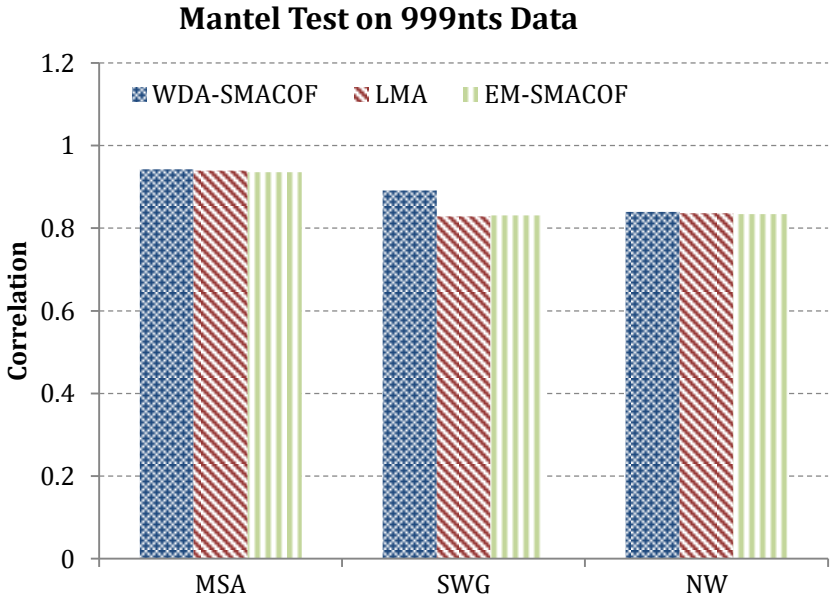

Figure 5.15: Mantel comparison of WDA-SMACOF, LMA and EM-SMACOF using distance input generated from one MSA method and two PWA methods on 999nts dataset

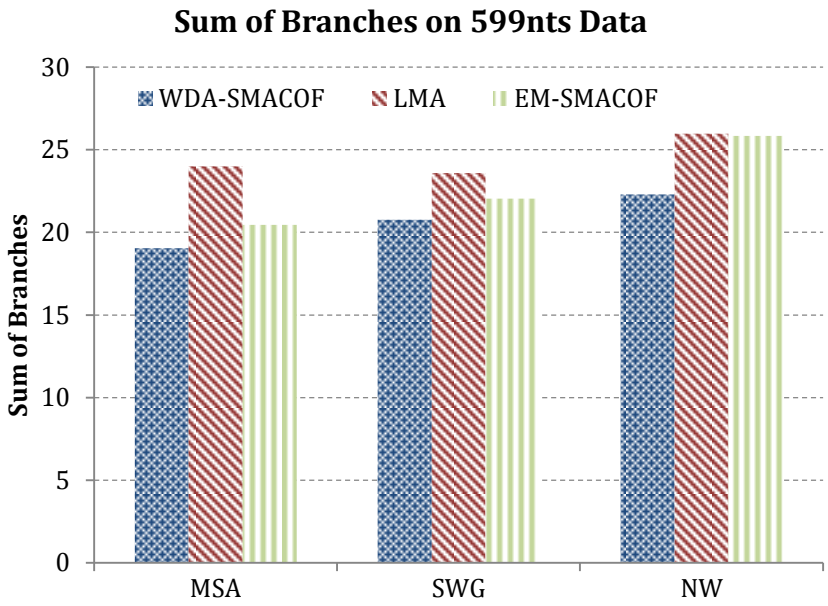

Figure 5.16: Sum of tree branches in 3D of WDA-SMACOF, LMA and EM-SMACOF using distance input generated from one MSA method and two PWA methods on 599nts dataset

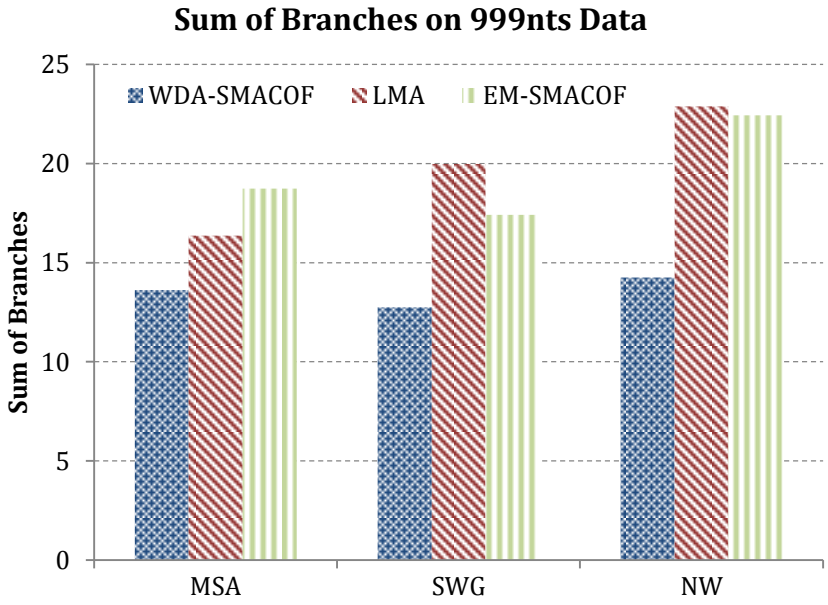

Figure 5.17: Sum of tree branches in 3D of WDA-SMACOF, LMA and EM-SMACOF using distance input generated from one MSA method and two PWA methods on 999nts dataset

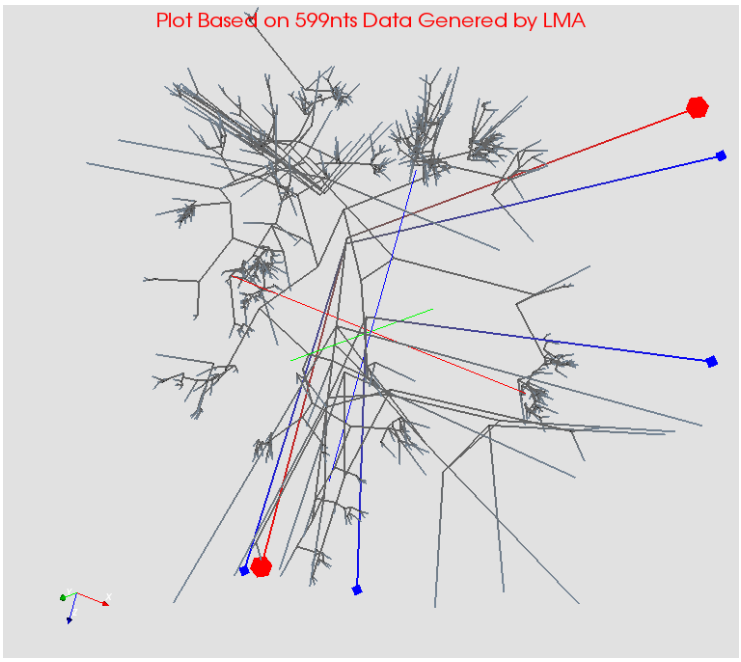

Figure 5.18: The plot of 599nts data using LMA MDS method on MSA distances. The red sphere points are the two highlighted points that are near each other from the phylogenetic tree. The blue square points are similar points that should belong to a same family.

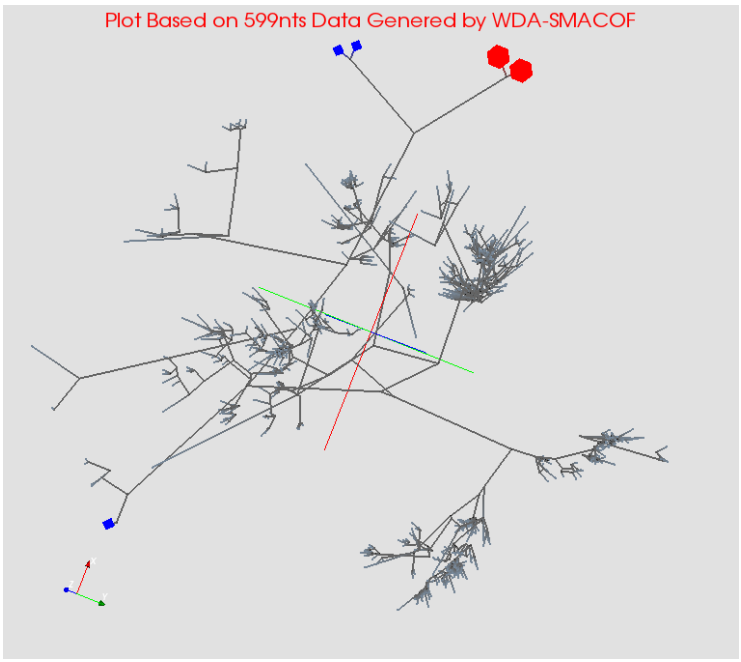

Figure 5.19: The plot of 599nts data using WDA-SMACOF method on MSA distances. The red sphere points are the two highlighted points that are near each other from the phylogenetic tree. The blue square points are similar points that should belong to a same family. And they are actually near each other.

5.5 Conclusion

In this section, two methods, referred to as Cuboid Cladogram and Spherical Phylogram, which enable display of clustering and phylogenetic tree simultaneously has been proposed. Cuboid Cladogram displays the clustering result on one side, and project the phylogenetic tree onto the locations of clustered points in 3D in order to achieve this. Spherical Phylogram directly uses the clustered points as the leaf nodes in the phylogenetic tree in order to construct the tree. Both methods are evaluated using intuitive observation in 3D space and statistic method during experiments. As input for MDS cluster visualization and phylogenetic analysis, we used sequences from the variable D2 domain of the 28S rRNA gene, which is commonly used for taxonomic identification of fungi . All sequences were from species of arbuscular mycorrhizal (AM) fungi because they exhibit a large amount of sequence variation both between species as well as within species, which can make them challenging to analyze. The sequence datasets were derived from a combination of: (1) a large-scale AM fungal phylogenetic study; (2) additional sequences obtained from GenBank to increase the taxonomic coverage of the dataset; (3) representative 454 pyrosequences from spores of known AM fungal species that were selected using DACIDR. DACIDR uses pairwise clustering and MDS for robust and scalable sequence clustering and visualization for more than one million sequences. The representative sequences are then selected from each cluster.

To compare the consistency between the clustering analysis and the phylogenetic tree, we implemented an algorithm we refer to as interpolative joining (IJ) in order to merge the traditional phylogenetic tree with the MDS cluster visualization into a spherical phylogram (SP). To evaluate how well the SP corresponded to the clustering result from the same dataset, we used a combination of the sum of branch lengths and Mantel tests in our experiments. The different experimental approaches generated similar results that show good agreement between the taxonomic delineations provided by the clustering and those provided by the phylogenetic analysis. This suggests that our proposed clustering technique based on pairwise alignment is a highly suitable alternative to phylogenetic analysis to study microbial communities.

Chapter 6. CONCLUSION AND FUTURE WORKS

6.1 Summary of Work

This dissertation mainly discussed about the optimization over MDS problem and extended the usage of that to phylogenetic analysis. The main purpose of the work is to make sequence data clustering and visualization a more scalable, reliable and widely application. The section 2 mainly discusses the background techniques mentioned in the dissertation. And a data clustering and visualization pipeline so called DACIDR is described since the optimization in this dissertation were based on the component in this pipeline. In section 3, an algorithm called WDA-SMAOF has been proposed in order to solve problem associate with sequence dissimilarity calculation. This is because the dissimilarities generated by using sequence alignment may have low quality issue so that it needs to be considered as missing. Furthermore, for special cases in sequence clustering, some sequences are more significant than other sequences, so the distances generated then should be given a more significant value. This algorithm has added a weight function support to a robust MDS algorithm called DA-SMACOF. Additionally, it uses conjugated gradient to avoid the additional time cost brought by this weighting function to this algorithm. The performance analysis shows that this WDA-SMACOF algorithm can scale well to large scale of processors and gives a more robust and accurate result than the existing methods. Then in Section 4, two

optimizations were done on the interpolation algorithm for the case with massive amount of sequence clustering and visualization by reducing the space and time cost. One optimization is to add weight function support to the data due to the same reason mentioned previously. This optimization so called W-MI-MDS can updated the original formula in order to achieve a higher accuracy result for sequence data with missing values. The other optimization, so called HE-MI, is to make the interpolation a hierarchical method, which greatly reduces the time cost of calculating the similarities between in-sample sequences and out-of-sample sequences. Finally in section 5, two new tree diagram, referred to as Cuboid Cladogram and Spherical Phylogram were proposed in order to determine the phylogenetic tree in a 3D space, and display the sequence clustering result along with phylogenetic trees, PCA is used to ensure the best result for Cuboid Cladogram and in order to achieve better result for Spherical Phylogram, the W-MI-MDS and WDA-SMACOF algorithms were used construct the tree diagram directly from the dimension reduction result.

6.2 **Conclusions**

Next generation sequencing (NGS) technique has high throughput of sequence generation. Massive amount of sequences can be generated within a short time from the gene samples with its help. Therefore, challenges have emerged about the method to do analysis on these sequences. To visualize the clustering result with large scale of sequences, it is essential to do customized optimizations for bioinformatics data on the multidimensional scaling (MDS) techniques. DACIDR is a pipeline uses pairwise clustering and MDS to do large scale sequence clustering and visualization. The optimization techniques proposed in this dissertation greatly improves the performances of DACIDR and extends its capabilities of processing millions of sequences along with phylogenetic analysis.

6.2.1 **WDA-SMACOF**

The WDA-SMACOF proposed in this dissertation has proposed a general solution for MDS problem with weighting data, as well as with the situation that part of data points needs to be fixed. The problem emerges because of the sequence alignment with bioinformatics data is not as

reliable as Euclidean distance calculation. And for special cases, various weights needs to be associated with each pair of sequence dissimilarities generated in sequence alignment. Traditional method on this type of problem requires cubic time complexity, but WDA-SMACOF proposed here avoided the high time cost of weighting function by using Conjugated Gradient (CG) instead of matrix inversion in traditional methods. In section 3, the detailed description of WDA-SMACOF is given, and how to apply CG on it is given in detail. The parallel version of WDA-SMACOF is written using iterative MapReduce framework, so the flowchart of it givens the detail about the work on main driver as well as the work for each mapper and reducer. Later in this section, the special version of WDA-SMACOF, called Fixed-WDA-SMACOF is given. The steps of how the final equation is listed as well as the algorithm description. Note that the parallelization of Fixed-WDA-SMACOF also uses the iterative MapReduce framework, and its parallelization involves a nested loop of 6 MapReduce jobs. The space and communication has been minimized on both of these parallel applications in order to run very large scale data visualization. In the experiments part, WDA-SMACOF gives a higher accuracy compared to all other algorithms with four different dataset, where the size of data varies from 4k to 400k. Not only the single thread version is tested, but also the parallel version has been tested on over 4000 cores. The parallel efficiency has been improved by adapting an iterative MapReduce framework called Harp over Twister. Furthermore, Fixed-WDA-SMACOF has been compared against an existing MDS interpolation method called MI-MDS. The result shows that it gives a more precise answer with the specific need of fixing part of the points. In short, WDA-SMACOF proposed in here gives a robust and scalable solution for dimension reduction on large scale of sequences by utilizing the power of iterative MapReduce and computer clusters.

6.2.2 **W-MI-MDS and HE-MI**

W-MI-MDS and HE-MI are two optimizations to the existing MDS interpolation algorithm proposed in section 4. W-MI-MDS is an algorithm that added weighting support for the out-of-sample problem, so that different weights can be associate with the distances from in-sample points to the out-of-sample points. HE-MI is a hierarchical method that proposed to reduce the time cost of MDS interpolation. This is needed because of the excessive time cost of sequence

alignment between the in-sample dataset and out-of-sample dataset. The detailed description of two tree structures, referred to as SSP-Tree and CN-Tree, used by this method is given in this section. The parallel versions of these two algorithms were similar. The out-of-sample problem is a task-independent problem, which means the applications solving it can be pleasingly paralleled. The flowchart of parallelization of these two algorithm using MapReduce framework is given. As iterative MapReduce framework also supports normal MapReduce application, so the parallel version of W-MI-MDS and HE-MI are implemented using Twister. In the experiments part of this section, both applications were tested and compared to other existing methods. W-MI-MDS gives a higher accuracy as well as a lower time cost with all dataset tested compared to the non-weighted MI-MDS. DA technique has also been analyzed in these experiments but did not show a significant different in terms of accuracy. HE-MI by using both SSP-Tree and CN-Tree has shown a much lower time cost compare to the MI-MDS data without missing distances. Overall, these optimization techniques has reduced the time cost for general MDS interpolation problem without missing distances or various weights requirement, and W-MI-MDS has provided a better solution than the MI-MDS by adding weighting function support to the data that has missing distances.

6.2.3 **Cuboid Cladogram and Spherical Phylogram**

Traditional phylogenetic tree analysis were solely based on the tree diagram generated using multiple sequence alignment and maximum likelihood method or other equivalent methods. Other the other hand, sequence clustering is usually done by using pairwise sequence alignment and hierarchical clustering or other equivalent methods. These two separate analysis on a same dataset could yield separate result with some variation. Thus sequence clustering could be complementary to phylogenetic analysis. Cuboid Cladogram and Spherical Phylogram are proposed in Section 5 to display sequence clustering result along with a phylogenetic tree, which enables the direct analysis on the same dataset with both tree diagram as well as the visualized clusters. In this section, Cuboid Cladogram is described in detail with the usage of principle component analysis (PCA). Since the sequence clusters are visualized in 3D by using MDS technique, a plane which all points are projected to needs to be selected. In order to keep the maximum variance of coordinates after the projection, PCA is used to select the plane. Then the points from the

clustering result are projected into the plane to be used as leaf nodes in the phylogenetic tree, which is determined by using independent phylogenetic analysis technique. Finally, the 3D constructed can be used for the final analysis and called Cuboid Cladogram. Spherical Phylogram is introduced later in Section 5. It uses interpolation to generate the tree. The points in the clusters are used directly for the phylogenetic tree instead of doing projection first, so the internal nodes from the phylogenetic tree needs to be interpolated into the 3D space. An algorithm called Interpolative Joining is proposed here. This algorithm can determine the coordinates for the internal nodes from given phylogenetic tree in the target dimension space (3D space), so the tree is generated as a spherical plot and the branch lengths are from the pairwise distances from the leaf nodes, thus the name Spherical Phylogram. The experiments in Section 5 illustrate the affection on the choice of sequence alignment to sequence clustering and phylogenetic tree construction. It shows a very high correlation between PWA and MSA in terms of generating the dimension reduction plot. Then the comparison among the choices of dimension reduction method is shown. It shows that WDA-SMACOF proposed in Section 3 has a best result of constructing spherical phylogram because it can avoid local optima during the optimization process. Generally speaking, Cuboid Cladogram and Spherical Phylogram can determine the phylogenetic tree in a 3D space with MDS clustering result, and enables the analysis on both results simultaneously.

6.3 Future Works

In this section, several possible improvements and research opportunities are discussed for interpolative MDS techniques. These improvements could benefit the data clustering and visualization pipeline (DACIDR) mentioned in Section 2.5.

6.3.1 Reduce Time Cost of WDA-SMACOF

Although the time complexity of WDA-SMACOF is $O(N^2)$, it is around 30 times longer processing non-trivial weights (Sammon's Mapping) than processing trivial weights (0s and 1s). Future study of the convergence threshold for CG could significantly impact the total time cost of WDA-SMACOF. One may use the target dimension mapping from previous SMACOF iteration to

set as starting position for CG in the next SMACOF iteration. This could potentially reduce the number of CG iterations needed per SMACOF iteration.

6.3.2 Hybrid Tree Interpolation

The methods for interpolation were using either CN-Tree or SSP-Tree separately. But these two trees could potentially be used together in order to achieve higher accuracy. SSP-Tree can clearly separate the in-sample points in the target dimension, but the distances from out-of-sample point to in-sample points used for interpolation are in high dimension, which causes bias for during the interpolation. CN-Tree were constructed in hyperspace to avoid this problem, but the points may be contained in multiple nodes range and being arbitrarily assigned to different tree node. A hybrid tree could construct the tree within hyperspace but clearly separates the points. Therefore, the accuracy of hierarchical interpolation result could be better.

6.3.3 Display Phylogenetic Tree with Million Sequence Clusters

As the phylogenetic tree were generated using reference sequences generated from half of million sequences, it will be interesting to display the phylogenetic tree within the original sequences clusters instead of just reference sequences. This way of displaying sequences could potentially find interesting clusters whose information cannot be fully represented within the representative sequence.

6.4 Contributions

As mentioned in Section 1.5, the contribution of this dissertation has been divided into 3 parts as the following:

1) **Robust and Scalable MDS with Weighting:** By leveraging the power of conjugated gradient, the time complexity of MDS with weighting can be reduced from cubic to quadratic, which makes it suitable for large scale dataset. And Deterministic Annealing technique is applied to avoid the local optima from the original algorithm. The experiments illustrated in Section 3 shows that the WDA-SMACOF is a reliable and scalable MDS algorithm that can outperform other existing dimension reduction methods. It successfully enables the visualization of nearly half of million sequences, and scale up to 4000 cores to process the dimension reduction.

2) **Hierarchical MDS Interpolation with Weighting:** The interpolation of MDS has been updated using a hierarchical algorithm called HE-MI instead of linear speed algorithm called MI-MDS. The time complexity is reduced to logarithmic from linear. The weighting function is added to support the missing values from in-sample points for a new algorithm W-MI-MDS. The experiments result shows that W-MI-MDS can always get a more accurate result with distances that has missing values, and reduces the time cost as well. HE-MI algorithm greatly reduces the time cost of MI-MDS by achieving approximate result for the out-of-sample problem.

3) **3D Phylogenetic Tree Displayed with Clusters:** This approach with Cuboid Cladogram and Spherical Phylogram can display the clusters in 3D with phylogenetic tree simultaneously. Therefore both analyses could be in viewed in 3D as well as it can be observed directly using naked eyes. By constructing phylogenetic tree directly from dimension reduction result, the clustering result could be displayed directly with the tree since clusters are naturally appeared during the MDS process. The MDS interpolation is used here for a new algorithm called Interpolative Joining to find the coordinates for the internal nodes in the tree structure of spherical phylogram. The experiments shows that by using WDA-SMACOF and pairwise sequence alignment, this method successfully identify the correlation of different AM fungal sequences families and proves its effectiveness.

BIBLIOGRAPHY

1. Petrosino JF, et al., *Metagenomic Pyrosequencing and Microbial Identification.* Clin Chem, 2009. **55**(5): p. 856-866.

2. Andersson AF, et al., *Comparative Analysis of Human Gut Microbiota by Barcoded Pyrosequencing.* PLoS ONE, 2008. **3**(7): p. e2836.

3. Cole JR, et al., *The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis.* Nucl Acids Res, 2005. **33**(suppl_1): p. D294-296.

4. Altschul, S.F., et al., *Basic Local Alignment Search Tool.* Journal of Molecular Biology, 1990. **215**: p. 403-410.

5. Hughes, A., et al., *Interpolative multidimensional scaling techniques for the identification of clusters in very large sequence sets.* BMC Bioinformatics, 2012. **13**(Suppl 2): p. S9.

6. Schloss, P.D., et al., *Introducing mothur: opensource, platform-independent, community-supported software for describing and comparing microbial communities.* Appl. Environ. Microbiol., 2009. **75**: p. 7537–7541.

7. Sun, Y., et al., *ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences.* Nucleic Acids Res., 2009. **37**(76).

8. Edgar, R.C., *Search and clustering orders of magnitude faster than BLAST.* Bioinformatics., 2010. **26**: p. 2460–2461.

9. Li, W. and A. Godzik, *Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences.* Bioinformatics, 2006. **22**: p. 1658–1659.

10. Geoffrey Fox, et al., *Case Studies in Data Intensive Computing: Large Scale DNA Sequence Analysis as the Million Sequence Challenge and Biomedical Computing*, in *Technical Report 9*2009.

11. Borg, I. and P.J. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. 2005: Springer.

12. Jolliffe, I., *Principal component analysis*. 2005: Wiley Online Library.

13. Bishop, C.M., M. Svensén, and C.K.I. Williams, *GTM: The generative topographic mapping.* Neural computation, 1998. **10**: p. 215--234.

14. Kohonen, T., *The self-organizing map*. Proceedings of the IEEE, 1990. **78**(9): p. 1464-1480.

15. Gannon, D., et al., *On Building Parallel & Grid Applications: Component Technology and Distributed Services.* Cluster Computing, 2005. **8**(4): p. 271-277.

16. *Sector and Sphere Data Intensive Cloud Computing Platform*. Available from: http://sector.sourceforge.net/doc.html.

17. Matei Zaharia, et al., *Spark: Cluster Computing with Working Sets*, in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10)*2010: Boston.

18. Grzegorz Malewicz, et al., *Pregel: A System for Large-Scale Graph Processing*, in *International conference on Management of data*2010: Indianapolis, Indiana, USA. p. 135-146.

19. J.Ekanayake, et al., *Twister: A Runtime for iterative MapReduce*, in *Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference June 20-25, 2010*2010, ACM: Chicago, Illinois.

20. Kearsley, A.J., R.A. Tapia, and M.W. Trosset, *The Solution of the Metric STRESS and SSTRESS Problems in Multidimensional Scaling Using Newton's Method*, 1995, Rice University: Houston, Tx.

21. Seung-Hee Bae, J.Y.C., Judy Qiu, Geoffrey C. Fox, *Dimension Reduction and Visualization of Large High-dimensional Data via Interpolation*, in *HPDC'10* 2010: Chicago, Illinois USA.

22. Seung-Hee Bae, et al., *Dimension reduction and visualization of large high-dimensional data via interpolation*, in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*2010, ACM: Chicago, Illinois. p. 203-214.

23. Cavalli-Sforza, L.L. and A.W. Edwards, *Phylogenetic analysis. Models and estimation procedures.* American journal of human genetics, 1967. **19**(3 Pt 1): p. 233.

24. Phipps, J., *Dendrogram topology.* Systematic Biology, 1971. **20**(3): p. 306-308.

25. Templeton, A.R., K.A. Crandall, and C.F. Sing, *A cladistic analysis of phenotypic associations with haplotypes inferred from restriction endonuclease mapping and DNA sequence data. III. Cladogram estimation.* Genetics, 1992. **132**(2): p. 619-633.

26. Page, R.D., *TreeView.* Glasgow University, Glasgow, UK, 2001.

27. Edgar, R.C., *MUSCLE: multiple sequence alignment with high accuracy and high throughput.* Nucleic Acids Res., 2004. **32**: p. 1792-1797.

28. Needleman SB, W.C., *A general method applicable to the search of similarities in the amino acid sequence of two proteins.* J Mol Biol, 1970. **48**: p. 443-453.

29. Gotoh, O., *An improved algorithm for matching biological sequences.* Journal of Molecular Biology, 1982. **162**(3): p. 705-708.

30. Smith, T.F. and M.S. Waterman, *Identification of common molecular subsequences.* Journal of Molecular Biology, 1981. **147**(1): p. 195-197.

31. Jeff Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. 2004 [cited 2010 November 6]; Presentation at OSDI-2004 Conference]. Available from: http://labs.google.com/papers/mapreduce-osdi04-slides/index.html.

32. *LINQ Language-Integrated Query*. 2009 [cited 2009 December]; Available from: http://msdn.microsoft.com/en-us/netframework/aa904594.aspx.

33. Hofmann, T. and J.M. Buhmann, *Pairwise data clustering by deterministic annealing.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1997. ***19***: p. 1--14.

34. Judy Qiu, et al., *Data Intensive Computing for Bioinformatics.* Technical Report, 2009.

35. Rose, K., E. Gurewitz, and G. Fox, *A deterministic annealing approach to clustering.* Pattern Recogn. Lett., 1990. **11**: p. 589--594.

36. Brusco, M.J., *A simulated annealing heuristic for unidimensional and multidimensional (city-block) scaling of symmetric proximity matrices.* Journal of Classification, 2001. **18**(1): p. 3-33.

37. Jan de Leeuw, *Applications of convex analysis to multidimensional scaling.* Recent Developments in Statistics, 1977: p. 133-145.

38. Moon, T.K., *The expectation-maximization algorithm.* Signal processing magazine, IEEE, 1996. **13**(6): p. 47-60.

39. Bae, S.-H., J. Qiu, and G.C. Fox. *Multidimensional Scaling by Deterministic Annealing with Iterative Majorization algorithm*. in *6th IEEE e-Science Conference*. 2010. Brisbane, Australia.

40. Gould, S.J., *Ontogeny and phylogeny*. 1977: Harvard University Press.

41. Schmidt, H.A., et al., *TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing.* Bioinformatics, 2002. **18**(3): p. 502-504.

42. Saitou, N. and M. Nei, *The neighbor-joining method: a new method for reconstructing phylogenetic trees.* Molecular biology and evolution, 1987. **4**(4): p. 406-425.

43. Ram, R. and M. Chetty, *MCMC Based Bayesian Inference for Modeling Gene Networks*, in *Pattern Recognition in Bioinformatics*. 2009, Springer. p. 293-306.

44. Ronquist, F. and J.P. Huelsenbeck, *MrBayes 3: Bayesian phylogenetic inference under mixed models.* Bioinformatics, 2003. **19**(12): p. 1572-1574.

45. Stamatakis, A., *RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models.* Bioinformatics, 2006. **22**(21): p. 2688-2690.

46. Ruan, Y., et al. *DACIDR: deterministic annealed clustering with interpolative dimension reduction using a large collection of 16S rRNA sequences*. in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*. 2012. ACM.

47. Krüger, M., et al., *Phylogenetic reference data for systematics and phylotaxonomy of arbuscular mycorrhizal fungi from phylum to species level.* New Phytologist, 2012. **193**(4): p. 970-984.

48. Benson, D.A., et al., *GenBank.* Nucleic Acids Research, 2012: p. gks1195.

49. Snir, M., et al., *MPI: The Complete Reference*. 1995, MA, USA.: MIT Press Cambridge.

50. Ruan, Y., et al. *Hymr: a hybrid mapreduce workflow system*. in *Proceedings of the 3rd international workshop on Emerging computational methods for the life sciences*. 2012. ACM.

51. Geoffrey Fox, *MPI and MapReduce*, in *Clusters, Clouds, and Grids for Scientific Computing CCGSC*2010: Flat Rock NC.

52. Kelley, C.T., *Iterative methods for optimization*. Vol. 18. 1999: Siam.

53. Bronstein, M.M., et al., *Multigrid multidimensional scaling*, in *Numerical Linear Algebra with Applications*2006, Wiley.

54. Mathar, R. and A. Žilinskas, *On global optimization in two-dimensional scaling.* Acta Applicandae Mathematica, 1993. **33**(1): p. 109-118.

55. Robinson, P.D. and A.J. Wathen, *Variational bounds on the entries of the inverse of a matrix*. IMA journal of numerical analysis, 1992. **12**(4): p. 463-486.

56. Dubois, P., A. Greenbaum, and G.H. Rodrigue, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors.* Computing, 1979. **22**(3): p. 257-268.

57. Ruan, Y. and G. Fox. *A Robust and Scalable Solution for Interpolative Multidimensional Scaling with Weighting*. in *eScience (eScience), 2013 IEEE 9th International Conference on*. 2013. IEEE.

58. Bulirsch, R. and J. Stoer, *Introduction to numerical analysis*. 2002: Springer Heidelberg.

59. Van der Vorst, H.A., *An iterative solution method for solving f(A)x=b, using Krylov subspace information obtained for the symmetric positive definite matrix A.* Journal of Computational and Applied Mathematics, 1987. **18**(2): p. 249-263.

60. Battiti, R., *First-and second-order methods for learning: between steepest descent and Newton's method.* Neural computation, 1992. **4**(2): p. 141-166.

61. Steele, J.M., *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. 2004: Cambridge University Press.

62. Stanberry, L., et al., *Visualizing the protein sequence universe.* Concurrency and Computation: Practice and Experience, 2014. **26**(6): p. 1313-1325.

63. Ruan, Y., et al., *Integration of Clustering and Multidimensional Scaling to Determine Phylogenetic Trees as Spherical Phylograms Visualized in 3 Dimensions.* Proceedings of C4Bio, 2014: p. 26-29.

64. Shewchuk, J.R., *An introduction to the conjugate gradient method without the agonizing pain*, 1994, Carnegie Mellon University, Pittsburgh, PA.

65. Bengio, Y., et al., *Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering.* Advances in neural information processing systems, 2004. **16**: p. 177-184.

66. Lu, Z., C. Sminchisescu, and M.Á. Carreira-Perpiñán. *People tracking with the laplacian eigenmaps latent variable model*. in *Advances in neural information processing systems*. 2008.

67. S. Xiang, F.N., Y. Song, C. Zhang and C. Zhang, *Embedding new data points for manifold learning via coordinate propagation.* Knowledge and Information Systems, 2009. **19(2)**: p. 159-184.

68. Priebe, M.W.T.a.C.E., *The Out-of-Sample Problem for Classical Multidimensional Scaling*, 2006, Indiana University: Bloomington, IN.

69. Cai, Y. and Y. Sun, *ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time.* Nucleic Acids Res., 2011. **39**(95).

70. Bentley, J.L., *Multidimensional binary search trees used for associative searching.* Communications of the ACM, 1975. **18**(9): p. 509-517.

71. Shevtsov, M., A. Soupikov, and A. Kapustin. *Highly Parallel Fast KD‐tree Construction for Interactive Ray Tracing of Dynamic Scenes*. in *Computer Graphics Forum*. 2007. Wiley Online Library.

72. Friedman, J.H., J.L. Bentley, and R.A. Finkel, *An algorithm for finding best matches in logarithmic expected time.* ACM Transactions on Mathematical Software (TOMS), 1977. **3**(3): p. 209-226.

73. Barnes, J. and P. Hut, *A hierarchical O (N log N) force-calculation algorithm.* 1986.

74. Berg, I. *Simulation of N-body problems with the Barnes-Hut algorithm*. 2009.

75. Zhang, T., R. Ramakrishnan, and M. Livny. *BIRCH: an efficient data clustering method for very large databases*. in *ACM SIGMOD Record*. 1996. ACM.

76. Tamura, K., et al., *MEGA6: molecular evolutionary genetics analysis version 6.0.* Molecular biology and evolution, 2013. **30**(12): p. 2725-2729.

77. Gouy, M., S. Guindon, and O. Gascuel, *SeaView version 4: a multiplatform graphical user interface for sequence alignment and phylogenetic tree building.* Molecular biology and evolution, 2010. **27**(2): p. 221-224.

78. Rambaut, A., *FigTree v1. 3.1: Tree figure drawing tool.* FigTree website, 2009.

79. Pons, J., et al., *Sequence-Based Species Delimitation for the DNA Taxonomy of Undescribed Insects.* Systematic Biology, 2006. **55**(4): p. 595-609.

80. Powell, J.R., et al., *Evolutionary criteria outperform operational approaches in producing ecologically relevant fungal species inventories.* Molecular Ecology, 2011. **20**(3): p. 655-666.

81. Fujisawa, T. and T.G. Barraclough, *Delimiting species using single-locus data and the generalized mixed yule coalescent (GMYC) approach: a revised method and evaluation on simulated datasets.* Systematic Biology, 2013: p. syt033.

82. Pavlopoulos, G.A., et al., *A reference guide for tree analysis and visualization.* BioData mining, 2010. **3**(1): p. 1.

83. Sanderson, M.J., *Paloverde: an OpenGL 3D phylogeny browser.* Bioinformatics, 2006. **22**(8): p. 1004-1006.

84. Munzner, T., *H3: laying out large directed graphs in 3D hyperbolic space*, in *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*1997, IEEE Computer Society. p. 2.

85. Pavlopoulos, G.A., et al., *Arena3D: visualization of biological networks in 3D.* BMC systems biology, 2008. **2**(1): p. 104.

86. Marco, A. and I. Mar ń, *A general strategy to determine the congruence between a hierarchical and a non-hierarchical classification.* BMC Bioinformatics, 2007. **8**(1): p. 442.

87. Abdi, H. and L.J. Williams, *Principal component analysis.* Wiley Interdisciplinary Reviews: Computational Statistics, 2010. **2**(4): p. 433-459.

88. Cardona, G., F. Rossello, and G. Valiente, *Extended Newick: it is time for a standard representation of phylogenetic networks.* BMC Bioinformatics, 2008. **9**(1): p. 532.

89. Choi, J.Y., et al., *PlotViz3: A cross-platform tool for visualizing large and high-dimensional data*, 2010.

90. Katoh, K. and M.C. Frith, *Adding unaligned sequences into an existing alignment using MAFFT and LAST.* Bioinformatics, 2012. **28**(23): p. 3144-3146.

91. Guillot, G. and F. Rousset, *Dismantling the Mantel tests.* Methods in Ecology and Evolution, 2013. **4**(4): p. 336-344.