DACIDR: Deterministic Annealed Clustering with Interpolative Dimension Reduction using Large Collection of 16S rRNA Sequences

Yang Ruan[1,3], Saliya Ekanayake[1,3], Mina Rho[2,3], Haixu Tang[2,3], Seung-Hee Bae[1,3], Judy Qiu[1,3], Geoffrey Fox[1,3]

Community Grids Laboratory[1]

Center for Genomics and Bioinformatics[2]

School of Informatics and Computing[3]

Indiana University Bloomington

{yangruan, sekanaya, mrho, hatang, sebae, xqiu, gcf}@indiana.edu

Abstract

The recent advance in next generation sequencing (NGS) techniques has enabled the direct analysis of the genetic information within a whole microbial community, bypassing the culturing individual microbial species in the lab. One can profile the marker genes of 16S rRNA encoded in the sample through the amplification of highly variable regions in the genes and sequencing of them by using Roche/454 sequencers to generate half to a few millions of 16S rRNA fragments of about 400 base pairs. The main computational challenge of analyzing such data is to group these sequences into operational taxonomic units (OTUs). Common clustering algorithms (such as hierarchical clustering) require quadratic space and time complexity that makes them not suitable for large datasets with millions of sequences. An alternative is to use heuristic methods; although it enables fast sequence analyzing, the hard-cutoff similarity threshold set for it and the random starting seed can result in reduced accuracy and overestimation. In this paper, we propose DACIDR: a parallel sequence clustering and visualization pipeline, which can address the overestimation problem along with space and time complexity issues as well as giving robust result. The pipeline starts with a parallel pairwise sequence alignment analysis followed by a deterministic annealing method of clustering and dimension reduction. No explicit similarity threshold is needed with the process of clustering. Experiments with our system also proved the quadratic time and space complexity issue could be solved with a novel heuristic method called Sample Sequence Partition Tree (SSP-Tree), which allowed us to interpolate millions of sequences with sub-quadratic time and linear space requirement. Furthermore, SSP-Tree can enhance the speed of fine-tuning on the existing result, which made it possible to recursive clustering to achieve accurate local results. Our experiments showed that DACIDR produced a more reliable result than two popular greedy heuristic clustering methods: UCLUST and CD-HIT.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering – *Algorithms;* C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Distributed Applications*;

General Terms

Algorithms, Performance

Keywords

Pairwise data clustering, multidimensional scaling, deterministic annealing, interpolation, exploratory data analysis

1. Introduction

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]. These methods can easily analyze small or medium sample sequences (e.g., with ten thousands of sequences) in order to allow scientists to draw meaningful conclusions. 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.

The first step of sequence analysis is typically generating sequences that representatives from the microbial community. One popular method is to use 16S rRNA sequences to study the phylogenetic relationship between different microbial species. Existing techniques to analyze such data are divided into two categories: taxonomy-based and taxonomy-independent [2]. Taxonomy-based methods provide classification information about the organisms in a sample. For example, BLAST [3] 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 16S rRNA 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 clustered them into OTUs by giving different threshold, such as 5% dissimilarity to determine if two sequences belong to a same genus or 3% dissimilarity to determine if they belong to same species. These methods doesn't require a pre-described reference database, thus they can enumerate novel pathogenesis as well as organisms in the preexisting taxonomic framework.

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 [8-11]. However, they can only perform on low dimensional data or lack accuracy.

Our techniques proposed in [12][13] for sequence analysis can be collectively classified as taxonomy-independent, wherein different sequence alignment tools are applied in order to glean specific pieces of information about the related genome. 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. An interpolation algorithm has been used to reduce time and space cost for massive data. All of these techniques are parallelized to process large data on multiple compute nodes, using MapReduce [14], iterative MapReduce [15] and/or MPI [16] frameworks. We improved the parallel efficiency of DACIDR by developing a hybrid workflow model on high performance computers (HPC) [17]. Additionally, we proposed SSP-Tree, which uses a heuristic method to achieve sub-quadratic time complexity with an interpolation process. Furthermore, we developed a new algorithm that can enable fast refinement of the clustering result by using SSP-Tree.

We describe the organization of the paper in the following: Section 2 discusses the background and related work. In Section 3 we describe DACIDR pipeline and various algorithms used in it. We present the SSP-Tree in Section 4. In Section 5 we show that choice over alignment methods is important, efficiency of SSP-Tree enabled interpolation and compared our results with two popular heuristic clustering methods. The conclusion and future work is presented in Section 6.

2. Related Work

There are already some taxonomy-independent heuristic or hierarchical methods exists in this area. MUCSLE+DOTUR is a popular pipeline for sequence analysis. MUCSLE [4] is used for multiple sequence alignment where it uses *k-mer* distance and a hierarchical method is applied to achieve fast speed. However, distance calculated from multiple sequence alignment might result in overestimation of the number of OTUs. So in our pipeline, we used pairwise sequence alignment instead of multi-sequence alignment. DOTUR [5] is one of the earliest hierarchical clustering methods for pyrosequencing and data analysis. DOTUR assigns sequences systematically to OTUs by using all possible distances. Therefore, a pairwise distance matrix must be generated as input for DOTUR. This causes its $O(N^2)$ time, disk space and memory complexity. So although it can generate reasonable result on small dataset, it can't be applied on massive data. HCLUST [6] is another similar method developed in mothur pipeline. Mothur is a well-known open-source, expandable software in the microbial ecology community. It is similar to a taxonomy-based clustering pipeline that a temporary pairwise distance matrix will be generated first by aligning input sequences against a pre-aligned reference database. Since generating a reference database is done before clustering, the computational complexity of the sequence-alignment step is $O(N)$ instead of $O(N^2)$. However, this algorithm will lead to inaccurate analysis to sequences from unknown microorganisms since there is no reference database for them. ESPRIT [7] is a method that tried to uses parallel computing to address the space and time issue in sequence analysis. It uses global pairwise alignment on each pair of sequences and the clustering method of it grouped sequences into OTUs on-the-fly, while keeping track of linkage information to overcome memory limitations. Although ESPRIT can experiment on hundreds of thousands of sequences, it has a time complexity of $O(N^2)$ thus has limitation on millions of sequences. ESPRIT-Tree [8] has been proposed later to address this issue. It uses probability sequences and a tree-like structure in hyperspace to reduce the time and memory usage for sequence analysis. Theoretically, it can reach quasilinear time complexity by inserting sequences through the root of the tree by only comparing the sequence and the center sequence in each tree node. Its tree construction relies on a subset of result from ESPRIT. Both of ESPRIT and ESPRIT-Tree uses *k-mer* distance for fast alignment between sequences, which has a high correlation with genetic distance. Although by using ESPRIT-Tree, sequence clustering has a time complexity of $O(NlogN)$, but the tree construction itself take $O(N^2)$ time, which can only be applied on small dataset.

Another direction to solve the taxonomy-independent clustering is greedy heuristic methods where several algorithms has been developed trying to solve this problem, such as CD-HIT [9], UCLUST [10] and AbundantOTU [11]. CD-HIT sorts the sequences first, and then the longest sequence becomes the representative of the first cluster. Each remaining sequence is compared with the representatives of existing clusters and assigned to an existing cluster or create a new cluster as the representative sequence based on the similarity. In each pair of sequences comparison, a short word filtering algorithm is used, which can determine if the similarity between two sequences is below a certain value without performing an actual sequence alignment. Therefore, by reducing the comparison times the actual computation time cost is saved as well. UCLUST uses a similar way of clustering compare to CD-HIT, but it can set a threshold of similarity below 80% while CD-HIT doesn't have this flexibility. Both of these two methods are capable of processing millions of sequences, however, the precision of their results suffer from the overestimation problem because a hard-cutoff similarity threshold is set and it's hard to tune this parameter for a reasonable clustering. Additionally, CD-HIT and UCLUST start the clustering by randomly giving the first sequence in a FASTA file to a new cluster as the reference sequence. Different from CD-HIT and UCLUST, AbundantOTU uses a consensus alignment algorithm to find the consensus sequence of each cluster without clustering them first so that its result is less affected by sequencing errors. Although this method can generate a clustering result better than CD-HIT on abundant species, it has a higher time complexity and lacks ability to group rare species correctly. In our pipeline, we proposed a deterministic annealing method of pairwise clustering, which can generate clusters automatically without having a threshold of similarity set or an initial seed. Clusters emerge as phase transitions as temperature is lowered [18]. This robust clustering method has been proved to be efficient over hundreds of thousands of sequences and indeed in many problem areas [19]. By using SSP-Tree method, we can process millions of sequences efficiently with a clustering result better than UCLUST and CD-HIT.

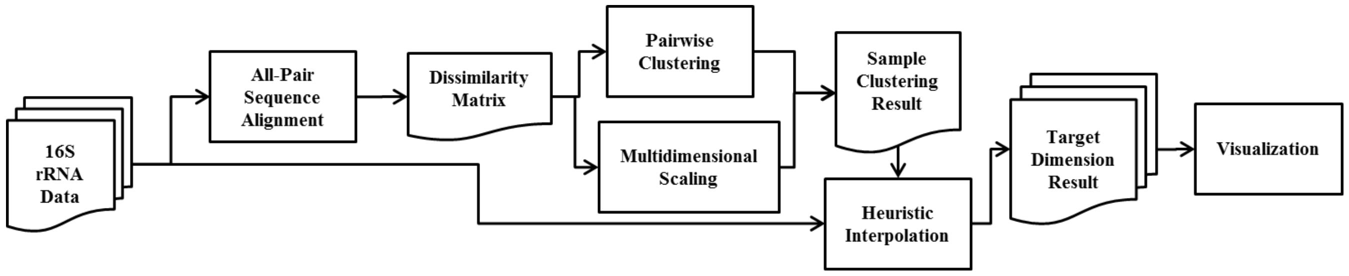


Figure 1 The structure of DACIDR pipeline

3. Data Clustering and Visualization Pipeline

As shown in Figure 1, DACIDR pipeline 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 generate a 3D mapping; Region Refinement is done on the PWC result along with the 3D mapping from MDS; Interpolation read the OTUs and plots to generate mapping for further sequences. In DACIDR, the 16S rRNA input dataset *D* is divided into a sample set *N* and an out-of-sample set *M*. The sample set *N* is processed at order N^2 by ASA, PWC and MDS, while out-of-sample set *M* is processed at order *M* by Interpolation. In this section, we will explain how the ASA, PWC, MDS and Interpolation work. Since the Region Refinement and Heuristic method of interpolation involves with SSP-Tree, they will be explained in next section.

3.1 All-pair Sequence Alignment

Biological similarity between two sequences is the property driving the DACIDR pipeline. Thus, to form a measurable value of similarity we first align the two sequences and compute a distance value for the 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 name all-pair sequence alignment.

In ASA, we chose Smith-Waterman (SW) [20] alignment method out of two well-known sequence alignment algorithms: Smith-Waterman and Needleman-Wunsch (NW) [21]. SW performs local sequence alignment and is more accurate than BLAST in this approach; that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, it compares segments of all possible lengths and optimizes the similarity measure. In contrast, NW performs a global alignment on two sequences which was not suitable for the particular dataset due to reasons mentioned under Section 5.1.

We used percentage identity to represent similarity among sequences, the distance δ between sequence i and sequence j is considered as the dissimilarity between them, which can be calculated in Equation 1:

$$\delta_{ij} = 1 - \frac{n'_{ij}}{n_{ij}} \qquad \text{Eq.1}$$

where n'_{ij} is the number of identical pairs between sequence i and sequence j and n_{ij} is the aligned sequence length.

SW 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 SW 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 made 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 [17].

3.2 Pairwise Clustering

As we use raw sequence data and not multiply aligned sequences, clustering is based on pairwise distances and must use appropriate algorithms. The deterministic annealing (DA) approach [22] introduced ~20 years ago for the vector spaces was modified ~10 years for pairwise case and extended by us to fully operational parallel software DA-PWC [12] using MPI. As noted above this approach is robust (inheriting the well-known advantages of annealing) and intrinsically multi-resolution. Temperature corresponds to pairwise distance scale and one starts at high temperature with all sequences in same cluster. As temperature is lowered one looks at finer distance scale and additional clusters are automatically detected from the appearance of negative eigenvalues for a second order derivative matrix first introduced by Rose [19] for vector clustering and extended by us to pairwise domain. We only need one parameter – namely the lowest temperature where one looks to split clusters; this corresponds to smallest size cluster desired. Other clustering methods like UCLUST and CD-HIT need more heuristic input.

To use DA-PWC in DACIDR, one inputs the dissimilarity matrix from ASA and outputs a group file, which contains the information about which cluster each sequence is assigned to.

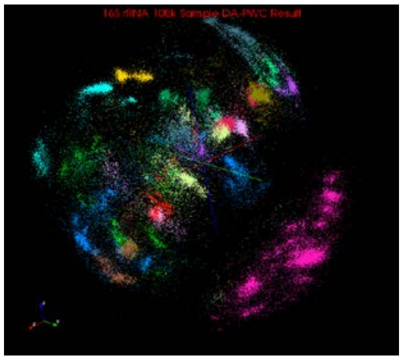
3.3 Multidimensional Scaling

Multidimensional scaling (MDS) is a set of related statistical techniques often used in information visualization for exploring similarities or dissimilarities in data. MDS algorithm use the pairwise distance matrix Δ and generate a mapping for each sequence to a point in an L-dimensional Euclidean space approximately preserving inter-point distances. Scaling by Majorizing a Complicated Function (SMACOF) algorithm is one of the MDS algorithms that have been proved to be fast and efficient [23][24]. It uses an Expectation Maximum (EM) method to minimize the objective function value, called Stress given in Equation 2.

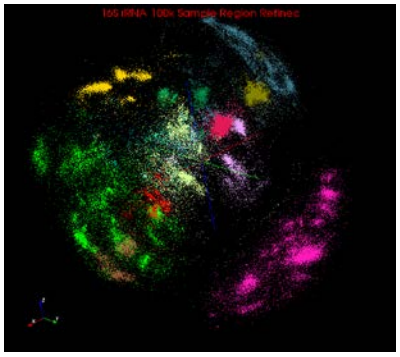
$$\sigma(X) = \sum_{i<j\leq N} w_{ij}\,(d_{ij}(X) - \delta_{ij})^2 \qquad \text{Eq.2}$$

where w denotes a possible weight, d_{ij} is the Euclidean distance from point i to j in the mapping and δ_{ij} is the distance from sequence i to j in **Δ.** However, it is well known that EM method suffers from local minima problem and we have developed a Deterministic Annealing (DA) enhancement to SMACOF with computational temperature [25].

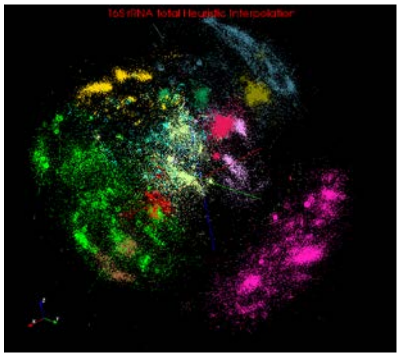
In DACIDR, we parallelized DA-SMACOF applications to make it usable for large sequences visualization by applying on iterative MapReduce paradigm. We set target dimension to 3 and



(a) Raw result from DA-PWC and DA-SMACOF on 100k sample sequences, 15 regions in total



(b) After region refinement on 100k sample sequences, 12 regions in total



(c) After interpolated 580k out-of-sample sequences by heuristic interpolation, 12 regions in total

Figure 2 Visualization result in 3D, each cluster is in different color (this is before final refinement)

visualized the mapping in a tool called PlotViz3 [26] that we developed. We call the 3D-coordinates result from MDS a plot, which can be integrated with the clustering result from PWC so that different clusters can be visualized in different colors/size/shape. In Figure 2(a), we have shown the raw result from PWC and MDS, where 15 clusters are generated with the 100k sample sequences selected from 16S rRNA dataset. Each sequence is mapped to a point in the 3D plot.

3.4 Interpolation

Although using DA on clustering and dimension reduction can generate robust result, both DA-PWC and DA-SMACOF have time(compute) and space(memory) complexity of *$O(N^2)$* which limits their applicability to large problems. Figure 5 illustrates that DA-MDS is applicable to other clustering algorithms. We will in a later paper describe how to improve DA-PWC performance to *$O(N)$* behavior in many circumstances. To overcome this difficulty, we adopted a technique called Majorizing Interpolation MDS (MI-MDS) [27], which is a simple interpolation approach based on pre-mapped MDS result of a sample set selected from the given data.

This algorithm's basic idea is to map out-of-sample data into target dimension space by k nearest neighbor *(k-NN)* interpolation without running full MDS on all of them. We added the function which can assign the out-of-sample data into designated cluster without running full PWC. Compare to existing MDS and PWC methods, this interpolation algorithm only need *$O(N)$* memory and time to execute. Furthermore, it's a pleasingly parallel application that it is highly efficient on multiple compute nodes. As described in following section we then divide full sample into regions and refine the clustering in small regions with computational modest cost.

4. SSP-Tree

In Section 3 we described the basic functionalities in the DACIDR pipeline. Although by using interpolation method, we made it possible to visualize and cluster millions of data, but the time complexity of MI-MDS algorithm remained high. The input dataset *D* is divided into sample set *N* and out-of-sample set *M*, so the computational complexity of MI-MDS remains *$O(N*M)$*, which means each sequence in the sample set will need to aligned with each sequence in the out-of-sample set. 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 1M 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.

To address the time complexity of MI-MDS, we used the concept from astrophysics simulations (solving *$O(N^2)$* particle dynamics) to split the sample data in *L*=3-dimension space into an octree with Barnes-Hut Tree (BH-Tree) [28] techniques. Our tree, called Sample Sequence Partition Tree (SSP-Tree) is similar to BH-Tree, and the sample dataset is divided up into cubic cells via an octree (in a *L*=3-dimension space), where the tree node set *K* is divided into two sets: each leaf node $e \in E$ will contain one sequence, and each internal node $i \in I$ will contain all the sequences belongs to its decedents. Each $i \in I$ has a child nodes set denoted as $\{C, 2^L\}$ where the number of its children smaller or equals to 2^L. Figure 3 is an example shown how the SSP-Tree works in 2D with 8 sequences. If a node contains only one sequence, then it became a leaf node; otherwise it is an internal node. In Figure 3, the node e_0 to e_7 contains the sequences from A to H accordingly. i_1 contains sequences A, B, C and D. i_2 contains sequences G and H. i_0 contains all the sequence as it is the biggest box.

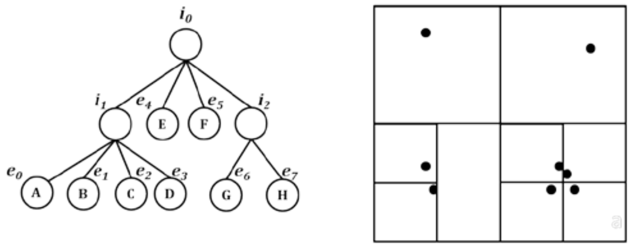


Figure 3 An example for SSP-Tree in 2D with 8 points

Algorithm 1: SSP-Tree Generation

```
Take every sample points in dimension $L$ space, take the $X_B^{max}$ and $X_B^{min}$ to construct the root node $B$.
For each sample $n$ in sample set $N$, insert it to node $k \in K$
    If $k$ doesn't has a sequence assigned, simply assign $n$ to $k$, and $k$ is added to $E$
    If $k$ belongs to $I$, determine $n$ should be inserted to $c$ in $\{C, 2^L\}$ of $k$ by comparing $X_n$ to $(X_k^{max} + X_k^{min})/2$. Insert $n$ to $c$.
    If $k$ belongs to $E$, remove the sequence $s$ assigned to $k$, insert $s$ to $\{C, 2^L\}$ of $k$; insert $n$ to $\{C, 2^L\}$ of $k$; $k$ added to $I$
```

A tree node can be represented in only two points in dimension *L*, which are $X_k^{max} = (x_0^{max}, x_1^{max}, x_2^{max}, \dots, x_L^{max})$ and $X_k^{min} =$

$(x_0^{min}, x_1^{min}, x_2^{min}, \dots, x_L^{min})$ where $k \in K$ and x_i^{max}, x_i^{min} denotes the maximum and minimum value of all the points's coordinates value in *L* dimensions. Constructing a SSP-Tree in *L*-dimension follows the procedure from Algorithm 1 where the only computation for it is to calculate the center of each node $k \in K$. Inserting the sample points into the tree doesn't need any extra computation where only comparison and assignment is needed, which makes SSP-Tree generation extremely fast. In our experiment, insert 100k sample points from 16S rRNA data into a SSP-Tree only takes about a few seconds on a desktop.

In SSP-Tree, every tree node *k* has a set of points P^k where P^B is the sample point set *N*. Each tree node *k* is represented by a center point p_c^k, which is the one nearest to the mass center inside each node. The mass center of node *k* is given by Equation 3

$$p_c^k = \{x_l^k \mid x_l^k = \sum_{i=0}^{n_k} \frac{x_l^i}{n_k}, 0 \le l < L\} \quad \text{Eq.3}$$

where n_k is the number of sequences in node *k*.

We describe a simple hierarchical majorizing interpolation method (HI-MI) as follows: One compares an out-of-sample point $\hat{p} \in \hat{P}$ to p_c^B first, and then recursively assign $\hat{p}$ to a nearest child node until the node contains nearest *k* neighbors is reached. This HI-MI method can reduce the time cost of interpolation from *O(N*M)* to *O(M*logN)*. However, its accuracy is poor due to the correctness of center point representation. It is obvious that the nodes in leaf set *E* are represents directly by the points they contained, so the representation is 100% accurate. But their parents in set *I* may contain several points, where could be in a same cluster or different clusters. 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 HI-MI method searches the tree from top to bottom, where starts with worst p_c, there is some probability that $\hat{p}$ could be assign to a different node other than the node the *k* nearest neighbors are in. To overcome this issue while keeping the lower time cost, we proposed a heuristic majorizing interpolation method (HE-MI).

```
Algorithm 2: Heuristic Majorizing Interpolation
Given a sample point set N, get a set of terminal nodes $T$ where point number in $t \in T$ is larger than a threshold μ where the number of regions $N^T \ll N$.
For each $\hat{p} \in \hat{P}$, compare the original distance δ between $\hat{p}$ and $p_c^t$ in $T$, assign it to the nearest node $t'$
All the sample points $p_0, p_1, p_2, \dots, p_{k'} \in P'_t$, in that node will be consider as the $k'$ nearest points to $\hat{p}$.
Find $k$ nearest points to $\hat{p}$. Compute every $\delta_{ij}$ between $\hat{p}_i$ and $p_j \in P'_i$;
Use the k-NN: $p_0, p_1, p_2, \dots, p_k \in P_t$, to $\hat{p}$. $(k \le k')$ to determine the position for $\hat{p}$ in dimension L. The group of $\hat{p}$ is assigned to the same group where the nearest $p_i$ is.
```

4.1 Heuristic Interpolation

First, we introduce the concept of terminal nodes *T* where $\{\boldsymbol{P_t} \mid \boldsymbol{t} \in \boldsymbol{T}\}$ is P_B. We can use optimization parameters, such as node level, maximum number of points inside, to control the number and quality of *T*. So instead of searching through top to bottom, we can directly use the high quality $\boldsymbol{p_c^t}(\boldsymbol{t} \in \boldsymbol{T})$ where *t* contains only one or few cluster inside to find nearest *k* neighbors for an out-of-sample point. Additionally, the number of *T* is much smaller than sample points number *N*. 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 2. By applying HE-MI, the time complexity is $O(MN^T)$. The time cost is greater than HI-MI, but the accuracy of interpolation is much higher in practice.

4.2 Region Refinement

Not only is SSP-Tree applied to dimension reduction and clustering so that it enables a fast and efficient way of interpolation, but also it can be used on fast refinement of existing DA-PWC result.

As we have clustering result from DA-PWC and mapping result DA-SMACOF, the clustering result can be refined using both of the factors. Here we call the raw clusters from DA-PWC mega-regions. After defining the mega-regions *g* in {*1…G*}, we classify the terminal nodes *T* into three categories: 1) Node cluster *g'* in *G'*, where a node cluster is assigned as the same cluster to the most points in that node. So the node in the node cluster actually represents the cluster of $P_{most}^{g'}$; 2) Unclear mixture *U*, where the unclear mixture is defined as a node contains significant number of points belonging to different clusters. As a terminal node may contain several different groups of points, this terminal node is undefined as to which *g* should it belongs to; 3) In the "intergalactic void" *V*, where normally the points inside these nodes are in between visually obvious clusters. Those points belong to *V* needs to be classified to clusters as well. Each terminal node *t* is represented by a center point p_c^t given in Equation 3. The goal of region refinement is to use the location information from MDS and the cluster information from PWC to classify node in {*1…G*} clearer and make region identification for nodes in *U*. Algorithm 3 describes region refinement process. To process with this algorithm, we set *f* as a cluster-define fraction threshold where cluster-define fraction is defined in Equation 4:

$$f_t^{g'} = \frac{n_t^g}{n_t} \qquad \text{Eq.4}$$

where n_t^g is the number of points in node *t* with assigned to *g*, and n_t is the total number of points in node *t*. We set a threshold *θ* as a number from 0.5 to 1. Node size, node level and number of points inside node are used in a node determination function *H* with a threshold *η* to distinguish the *V* from *U* and *G'*.

```
Algorithm 3: Fast Region Refinement
Iterate Following
  Create SSP-Tree and get $T$
  Loop over $t \in T$
    If $H(t) < \eta$, t is added to set V
    If $H(t) \ge \eta$,
      If $f_t^{g'} > \theta$, assign t to g and t is added to set G.
      If no $f_t^{g'} > \theta$ $(g \in G)$, t is added to set U.
  Loop over $t \in T$
    Update center point $p_c^t$
  Loop over p in $t \in U \cup G$
    Assign p to g where distance$(p, p_c^{\hat{t}})$is minimum and $\hat{t} \in G$
  If all $p_c^{\hat{t}}$ in $\hat{t} \in U$ are the same in last iteration, break
  Else, continue
Finally assign all $p \in P$ to $\hat{t} \in G$ where distance$(p, p_c^{\hat{t}})$ is minimum
```

After the region refinement, the cluster with high density near each other can be merged automatically, and the cluster with lower density can be reassigned with more points. By observing from the plot with the region refinement result and raw DA-PWC

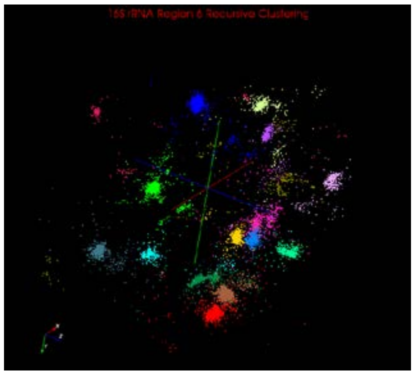


Figure 4 Recursive clustering result for mega-region 6 in DACIDR result of whole dataset

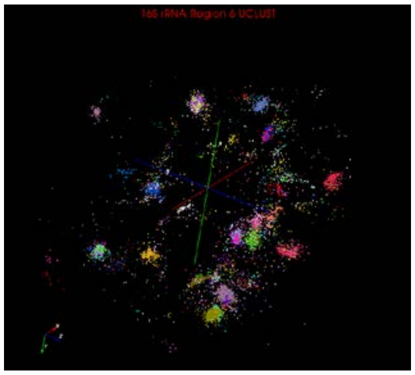


Figure 5 UCLUST result for mega-region 6 in DACIDR result of whole dataset

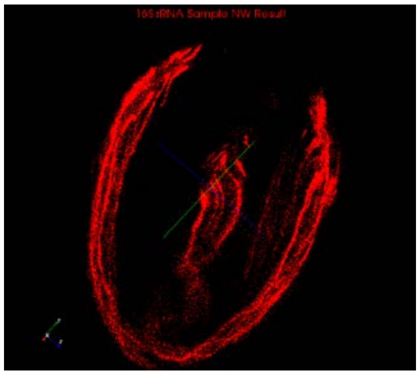


Figure 6 Visualization result for 100k sample set using NW distance

result, our mega-regions are much clearer as shown in Figure 2(b). Region 9(dark grey), 12(purple) and 15(light green) on the upper right of Figure 2(a) have been refined and merged into one region(grey). Region 8(light blue) on the top left is split and becomes part of cluster 2(green) and 4(yellow). Furthermore, this method is extremely fast since it the number of terminal nodes is much smaller than *N*. The computational cost of algorithm 3 is very small that it takes about 10 seconds to process a 100k dataset on a desktop.

4.3 Recursive Clustering

By applying HE-MI to the result from region refinement on 100k sample data, all the sequences from hmp16S rRNA data have been successfully clustered and visualized as shown in Figure 2(c). However, because each of these clusters contains several hundreds of thousands sequences, they still have internal structures which seems 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. Therefore, to amplify the dissimilarity between sub-clusters, we introduce the recursive clustering, which is to apply DACIDR on each separate region. The recursive clustering result of region 6(dark green) in shown in Figure 4. 16 clusters were found in this region which shows clear separation between each cluster.

5. Experiments

The experiments were carried out on PolarGrid (PG) cluster using 100 compute nodes and Tempest using 32 compute nodes. The compute nodes we used on PG are iDataPlex dx340 rack-mount servers with 8 cores per node. Tempest is an HP distributed shared memory cluster with 768 processor cores. The data was selected within 16S rRNA data from the NCBI database. The total input sequence number is 1160946. First, we examined the dataset and found all duplicate sequences, which have exactly the same length and composition. Then we screened the data by keeping only one sequence in each duplicate group, so that every sequence in the filtered set is different from each other. Finally, we could have a unique data set of 684769 sequences. Since the rest of the sequences all have a corresponding unique sequence in the unique set, they can be assigned to clustering result directly.

5.1 SW versus NW

We evaluated both SW and NW on the sample *N=100k* dataset before proceeding with the rest of the pipeline and found SW to produce more reliable results than NW. Sequence lengths were not uniform in the 16S 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 we computed 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 6, which later we identified as a direct consequence of the number of gaps present in the alignment. Pictorially, this effect is shown in Figure 7. 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.

The analysis of the line structure is shown in Figure 8, which concludes that points along the line are linearly related in lengths and NW has introduced gaps linearly to form global alignments. The sequences from 2-9 are aligned with Sequence 1, whose length is the longest. It shows that original lengths decrease linearly from one end to the other. The mismatches introduced by gaps for the alignments of these sequences have thus increased linearly according to the Mismatches by Gaps line. Also, clear is the fact that gaps have a dominant effect on the number of mismatches as the Total Mismatches line overlaps with the Mismatches by Gaps line. Thus, aligning short sequences with long sequences using NW has caused the introduction of biologically unimportant number of gaps purely for the sake of forming a global alignment.

SW in contrast performed a local alignment producing alignment segments with fewer gaps. This reduction in superfluous gaps immediately improved the quality of clustering and plots where more globular structure was evident rather long thin cylinders.

5.2 Comparison with UCLUST and CD-HIT

We have used two popular choices of clustering methods: UCLUST and CD-HIT to compare the result with DACIDR. As mentioned in previous section, UCLUST and CD-HIT are two popular greedy heuristic methods which are capable of processing millions of sequences on a desktop. Thus we apply these two methods on our dataset.

From Figure 9 it is shows that by directly applying CD-HIT or UCLUST on the whole 16S rRNA dataset we have, the clustering

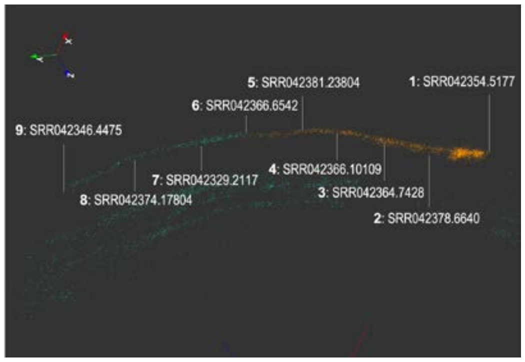


Figure 7 Long thin formation of points resulting from NW alignment (Point ID Number: Sequence ID)

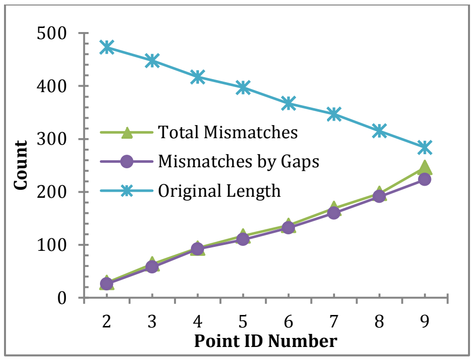


Figure 8 Effect of gaps towards the long thin structure

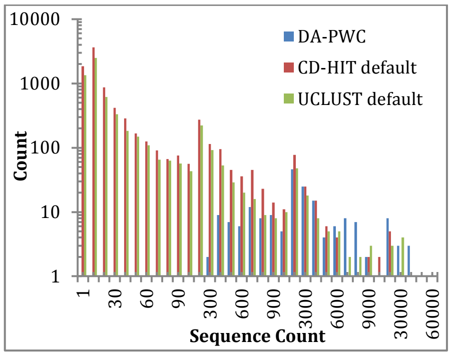


Figure 9 Histogram of number of clusters found based on number of sequences in each cluster

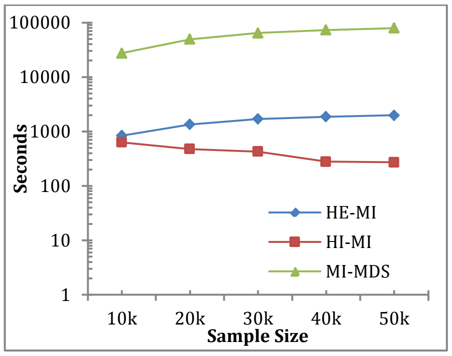


Figure 10 Execution time of three interpolation method

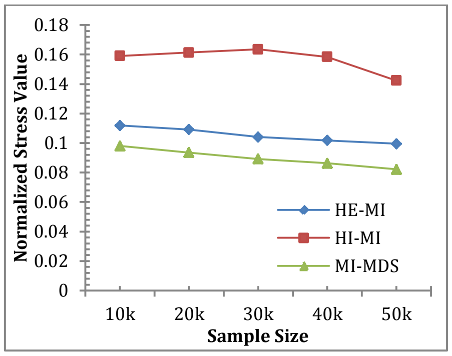


Figure 11 Normalized Stress value of 100k interpolation mapping result

result is overestimate. By using DACIDR on the whole dataset and one more time on each region, a total number of 188 clusters are found, and they contain a reasonable number of sequences in each cluster, from 300 to 40000. However, by using CD-HIT and UCLUST with a dissimilarity of 0.97, we found 8418 and 6000 clusters. Among the clusters found, most of them only contain 1 or 2 sequences. As shown in the histogram, CD-HIT found 5475 clusters only have less than 10 sequences in them, and UCLUST found 3829 such clusters. And if we lower the dissimilarity threshold to 0.90 for both of the methods, some cluster contains over 100000 sequences will be found along with many clusters still have one or two sequences inside. Figure 5 is the visualization result we used to show how UCLUST works as different color for each cluster. The UCLUST results are messier and single clusters are broken into several components. The inaccuracy of both these two methods happens because of two reasons: one is the hard-cutoff dissimilarity threshold where the optimized value is difficult to determine for a large input dataset; the other one is both of these methods use global alignment, which can result in unreliable answer for a dataset where sequences have very different lengths, which in our case, is the 16S rRNA dataset. This experiment demonstrates that DACIDR can have a robust clustering result which is better than CD-HIT and UCLUST. DACIDR is computationally more complicated but we have shown how using interpolation and SSP-Tree, we get practical computation and memory requirements.

5.3 Comparison of Interpolation Methods

In this experiment, we conduct three interpolation methods compare with each other in execution time and normalized stress value which is given in Equation 5:

$$\sigma(X) = \sum_{i<j\leq N} \frac{1}{\sum_{i<j}\delta_{ij}} (d_{ij}(X) - \delta_{ij})^2 \qquad \text{Eq.5}$$

where the notations are from Equation 2. Generally speaking, the normalized stress value is the error value from target dimension mapping to the original dimension. Therefore, a mapping result has a higher accuracy when the normalized stress value is lower.

This test is done using the 100k dataset from 16S rRNA data on 32 nodes from PG. We selected 10k, 20k, 30k, 40k and 50k from it as sample sets and the rest 90k, 80k, 70k, 60k and 50k are considered as out-of-sample sets. The sample sets are assumed to have the mapping in target dimension.

Figure 10 shows that HE-MI and HI-MI execute interpolation much faster than MI-MDS while both of former methods takes around 1000 seconds to finish and MI-MDS takes about 50 times longer than that. The computation for MI-MDS is *O(MN)* where *N* is the sample size and *M* is the out-of-sample size. Note that both HE-MI and MI-MDS's execution time increases while out-of-sample size decreases. This is because computation for both of these methods correlates with sample size * out-of-sample size while this value increases from 10k * 90k to 50k * 50k. But for HI-MDS, since it's time complexity is *O(MlogN)*, so *logN* will remains almost same from *N* increases from 10k to 50k. And *M* decreases from 90k to 50k, so its execution time decreases. Figure 11 shows that MI-MDS has the most accurate result because of computing every distance between each sample and out-of-sample point. However, this experiment shows that by using HE-MI, the interpolation processes much faster than MI-MDS, and the accuracy of the mapping result is much better than HI-MI, which makes HE-MI the ideal solution for massive size of data interpolation.

6. Conclusion and Future Work

In this paper we proposed a parallel data clustering and visualization method: DACIDR, which can efficiently cluster millions of sequences with various lengths. DACIDR utilizes the computing power of HPC by applying on several distribute and parallel computing frameworks. Compared to traditional sequence clustering method without visualization, such as UCLUST and CD-HIT, our visualization result combined with the clustering result can help biologist observe and analysis structures among different gene clusters. These correlations enable us to cluster millions of sequences efficiently with high accuracy. Using the deterministic annealing method can help us avoid local optima and overestimation problem. By using SSP-Tree in DACIDR, not only can the interpolation to clustering and visualization result run faster, but also we can refine the result from DA-PWC for hundreds of thousands results in a few seconds.

We are currently integrating phylogenetic trees with our analysis both by adding it to visualization and using it to improve specification of mega-regions where there are ambiguous clusters.

7. Acknowledgments

Our thanks to UITS in Indiana University for Polar Grid support and Ryan Hartman from CGL in Indiana University for Windows HPC cluster support. This work is under National Health Institute Grant 1RC2HG005806-01 support.

8. References

[1] Peterson J, Garges S, et al. (2009). "NIH Human Microbiome Project." Genome Research. **19**(12): 2317-2323.

[2] Cole JR, Chai B, et al. (2005). "The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis." Nucleic Acids Res. **33**(suppl_1): D294-296.

[3] Altschul, S. F., et al. (1990). "Basic Local Alignment Search Tool." Journal of Molecular Biology. **215**: 403-410.

[4] Edgar, R. C. (2004). "MUSCLE: multiple sequence alignment with high accuracy and high throughput." Nucleic Acids Res. **32**: 1792-1797.

[5] Schloss, P. D. and J. Handelsman. (2005). "Introducing DOTUR, a computer program for defining operational taxonomic units and estimating species richness." Appl. Environ. Microbiol. **71**: 1501-1506.

[6] Schloss, P. D., S. L. Westcott, et al. (2009). "Introducing mothur: opensource, platform-independent, community-supported software for describing and comparing microbial communities." Appl. Environ. Microbiol. **75**: 7537–7541.

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

[8] Cai, Y., et al.(2011). "ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time." Nucleic Acids Res. **39**(95).

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

[10] Edgar, R. C. (2010). "Search and clustering orders of magnitude faster than BLAST." Bioinformatics. **26**: 2460–2461.

[11] Yuzhen Ye. "Identification and quantification of abundant species from pyrosequences of 16S rRNA by consensus alignment." The Proceedings of BIBM 2010, 153-157

[12] Fox, G. C. (2011). "Deterministic Annealing and Robust Scalable Data Mining for the Data Deluge." PDAC'11, Seattle, Washington, ACM.

[13] Hughes, A., Y. Ruan, et al. (2012). "Interpolative multidimensional scaling techniques for the identification of clusters in very large sequence sets." BMC Bioinformatics 13(Suppl 2): S9.

[14] Dean, J. and S. Ghemawat (2008). "MapReduce: simplified data processing on large clusters." Commun. ACM 51(1): 107-113.

[15] J.Ekanayake, et al. "Twister: A Runtime for iterative MapReduce." Proceedings of MapReduce'10 of ACM HPDC 2010, Chicago, Illinois, ACM.

[16] Snir, M., S. Otto, et al. (1995). "MPI: The Complete Reference." MA, USA, MIT Press Cambridge.

[17] Ruan, Y., Z. Guo, et al. "HyMR: a Hybrid MapReduce Workflow System." Proceedings of ECMLS'12 of ACM HPDC 2012, Delft, Netherlands, ACM.

[18] Rose, K., Gurewitz E., Fox, G. C. (1990). "Statistical mechanics and phase transitions in clustering." Phys. Rev. Lett. **65**: 945--948.

[19] Rose, K. (1998). "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems." Proceedings of the IEEE **86(11)**: 2210--2239.

[20] O. Gotoh, (1982) "An improved algorithm for matching biological sequences." Journal of Molecular Biology. 162:705-708.

[21] Needleman, Saul B. and Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." Journal of Molecular Biology 48 (3): 443–53.

[22] Rose, K., Gurewwitz, E., and Fox, G. (1990). "A deterministic annealing approach to clustering." Pattern Recogn. Lett.11: 589-594.

[23] Bronstein, M. M., A. M. Bronstein, et al. (2006). "Multigrid multidimensional scaling." Numerical Linear Algebra with Applications. Wiley.

[24] Borg, I., and Groenen, P. J. (2005) "Modern Multidimensional Scaling: Theory and Applications." Springer, 2005.

[25] Bae, S.-H., J. Qiu, et al. (2010). "Multidimensional Scaling by Deterministic Annealing with Iterative Majorization algorithm." Proceedings of the 6th IEEE e-Science Conference, Brisbane, Australia.

[26] PlotViz - A tool for visualizing large and high-dimensional data. http://salsahpc.indiana.edu/pviz3/

[27] Bae, S.-H., J. Y. C., et al. (2010). "Dimension reduction and visualization of large high-dimensional data via interpolation." Proceedings of the 19th ACM HPDC Conference, Chicago, Illinois, ACM.

[28] J. Barnes and P. Hut (1986). "A hierarchical O(N log N) force-calculation algorithm." Nature **324** (4): 446–449