2019

Scalable Clustering for Immune Repertoire Sequence Analysis

Prem Bhusal
Wright State University

SCALABLE CLUSTERING FOR IMMUNE REPERTOIRE SEQUENCE ANALYSIS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

by

PREM BHUSAL
B.E., Visvesvaraya Technological University, 2014

2019
Wright State University

Wright State University
GRADUATE SCHOOL

April 23, 2019

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Prem Bhusal ENTITLED Scalable Clustering for Immune Repertoire Sequence Analysis BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science .

Keke Chen, Ph.D.
Thesis Director

Mateen M. Rizki, Ph.D.
Chair, Department of Computer Science

Committee on
Final Examination

Keke Chen , Ph.D

Krishnaprasad Thirunarayan (T.K.Prasad) , Ph.D

Tanvi Banerjee, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Bhusal, Prem. M.S., Department of Computer Science and Engineering, Wright State University, 2019. *Scalable Clustering for Immune Repertoire Sequence Analysis.*

The development of the next-generation sequencing technology has enabled systems immunology researchers to conduct detailed immune repertoire analysis at the molecule level. Large sequence datasets (e.g., millions of sequences) are being collected to comprehensively understand how the immune system of a patient evolves over different stages of disease development. A recent study has shown that the hierarchical clustering (HC) algorithm gives the best results for B-cell clones analysis - an important type of immune repertoire sequencing (IR-Seq) analysis. However, due to the inherent complexity, the classical hierarchical clustering algorithm does not scale well to large sequence datasets. Surprisingly, no algorithms have been developed to address this scalability issue for immunology research. In this thesis, we study two different strategies, aiming at finding the best scalable methods that can preserve the quality of hierarchical clustering structure. The two strategies include (1) non-Euclidean indexing methods for speeding up the classical hierarchical clustering(HC), (2) a new tree-based sequence summarization approach - SCT that scans the large sequence dataset once and generates summaries for hierarchical clusters(HC). And we also experimented with the Spark based minimum-spanning-tree algorithm (SparkMST) that generates the equivalent result of single linkage hierarchical clustering (SLINK) for comparative analysis.

We have implemented all these algorithms and experimented with real sequence datasets for B-cell clones analysis. The result shows that (1) the indexing-enhanced HC (e.g., using the Vantage-Point tree for indexing) preserves the clustering quality very well, while also significantly reducing the time complexity of the original HC; (2) SCT with HC is the fastest approximate HC method with slightly sacrificed quality; and (3) SparkMST scales out satisfactorily and gives significant performance gain with a large Spark cluster.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to my thesis adviser Dr.Keke Chen for supporting me throughout my journey in the Graduate school. I would also like to thank my thesis committee members, Dr.Krishnaprasad Thirunarayan (T.K.Prasad) and Dr.Tanvi Banerjee for taking the time to review my thesis.

I would also like to thank all my friends for their support and making my experience at graduate school such a wonderful one.

Last but not least, I would like to thank my mother, my sister and my brother-in-law for their continuous support throughout this journey.

Introduction

Clustering is basically grouping of the similar object together from huge dataset so that object belonging to one cluster (group) are closely related to each other than the object present in other cluster. So the main task in clustering is to partition the huge set of data into a small closely related group where the closeness is defined with some similarity criteria (e.g. distance). Clustering is unsupervised learning process where input data is provided without the previous knowledge of inherent structure of group or class in the data and the clustering method should identify most appropriate organization of groups without any external information. It is a popular technique for data analysis in various field such as data mining,machine learning, bioinformatics,image processing , information retrieval and so on.

Fine clustering is one of the challenging problems because the complexity of finding final cluster structure increases as the dimension and size of input data increases.

Hierarchical Clustering(HC) is one of the widely used methods in the bio-medical field for gene expression data analysis, genomic sequence analysis and so on. In this thesis, we study the hierarchical clustering for the analysis of Immune Repertoire-Sequence data.

1.1 Motivation

The human immune system works amazingly well in defending the body against attacks by “foreign” invaders. One of the important mechanisms is *adaptive immunity* that can

respond to many types of infections automatically. The core of this mechanism is that the B cells, a subtype of white blood cells, can adaptively generate a variety of antibodies that fight a different type of antigens, e.g., bacteria or viruses. Without infection, the B-cells already carry genes for generating about $\sim 10^7$ unique immunoglobulin (Ig) molecules that determine the types of antibodies. Upon activation (e.g., infection), these naive B-cells will further diversify through a process called somatic hypermutation to better handle new types of antigens. It has been observed that in healthy human adults, about 7 % of B cells are mutated [28], which means about 10^6 new types of Ig molecules may be generated by each activation. The mutation rate and the B-cell clone distribution may vary due to aging [27] and disease [7, 17]. Profiling a person's B-cell clones can help us understand the healthiness level of their immune system.

Profiling B-cell clones at the molecule level was an impossible mission until the next-generation sequencing technique [22, 20] was invented, which is often referred as the immune repertoire sequencing (IR-seq) approach. The next-gen technique can profile such a large scale of mutations at a low cost. For example, Illumina MiSeq costs only about $500 for sequencing one gigabase pairs (one gigabase pairs (Gb) = 10^9 base pairs). A typical sequence for studying B-cell mutation has about 100-300 base pairs and one experiment may generate about millions of sequences. Thus, IR-seq has become a popular and economical tool for molecule-level immune systems research, including B-cell clones analysis.

The diversity of B-cell comes from the unique VDJ recombination process (Figure 1.1) which produces millions of B-cells that each bear a unique antibody gene. B-cells randomly assemble different gene segments - known as the variable (V), diversity (D) and joining (J) genes, to generate unique antibodies. Antibody repertoire is able to further diversify upon antigen stimulation through somatic hypermutation (SHM), which randomly mutates the antibody genes. As illustrated in Figure 1.1, single original ancestral nave B cell could become several different subpopulations, each with different SHM pattern and cell abundance. The IR-seq method profiles this VDJ segment (200-300 base pairs)

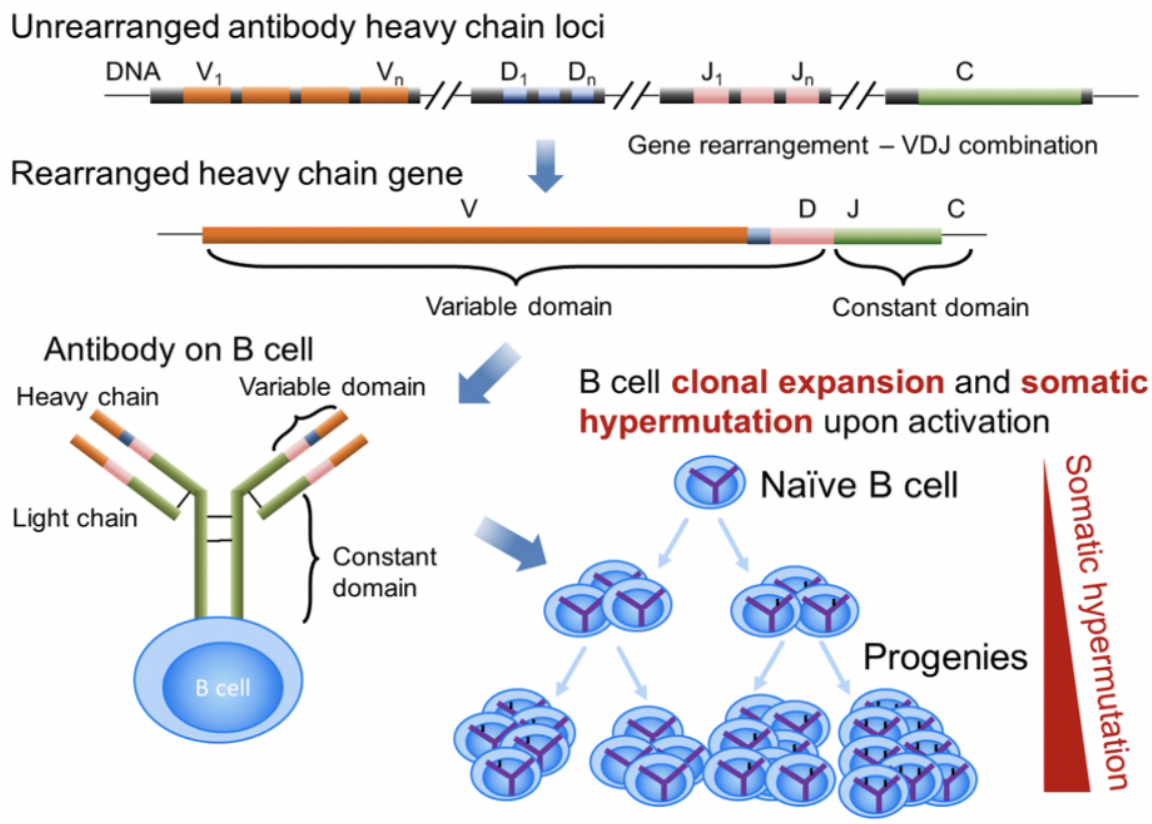

Figure 1.1: Antibody repertoire generation and diversification during activation

to understand the diversity and similarity between mutations as the mutation happens in the VDJ junction region. To identify the B-cell clones, one effective method is to study the clustering structures of VDJ junction segments in one person's B cells, e.g., extracted from his/her blood samples. In particular, the hierarchical clustering structure gives good clues about the B cell evolving patterns - how one VDJ mutates into another. Gupta et al. [12] has shown that among hierarchical clustering algorithms, the single-linkage algorithm (SLINK) gives the best results for B-cell clones analysis. Recently, we have also applied this clustering analysis method to understand the patterns of B-cell clones and how they evolve for malaria patients [27]. The key challenge of this clustering analysis is the number of profiled sequences can be very large from hundreds of thousands to millions. However, the optimized SLINK clustering methods still take $O(N^2)$ time complexity for N sequences - in practice, with a classical implementation running on a single machine, it may take

days to get the clustering results for merely hundreds of thousands of sequences, which is inconvenient for domain experts. With the increased scale of studies that will generate more sequences, any significant development of faster and scalable single-linkage (and hierarchical, in general,) clustering algorithms will benefit the whole systems immunology research community, and possibly other biomedical research domains [4], as well, that use clustering of large-scale sequence data as a major analytic tool.

We have developed a complete immune repertoire sequencing (IR-Seq) analysis framework and used it in several applications [27, 13]. Fig 1.2 gives the overall structure and different components of IR-seq analysis framework. It consists of four stages namely (1) Sequence Reads Processing, (2) Sequence Annotation, (3) Lineage Formation, and (4) Analyses. The algorithms used in the stages (1) and (2) have linear time complexity and they scale well with large datasets with naturally parallel processing [8, 9]. Reads processing has some well-known modules, such as read 1 and 2 alignment for paired-end sequencing, barcode removal, consensus building, and sequence truncating, which we skip the details[27]. Typically, the algorithms either work on pairs of reads or individual sequences and simple linear complexity algorithms (e.g., key-based aggregation for consensus building). Similarly, sequence annotation works on individual sequences. It's straightforward to scale up the processing with simple models like MapReduce [8]. After Lineage Formation, the lineage-level analyses [5] involve only thousands or tens of thousands of lineages, where scalability and speed become less serious for the common analytics algorithms.

Among all the components, Lineage Formation uses a clustering algorithm to find groups of similar sequences and then generate the lineage, i.e., the representative sequence for each group. It has become the bottleneck for large sequence datasets (in range of hundreds of thousands to millions) which we will discuss in detail in the problem definition section. Our main focus is to develop a fast and scalable hierarchical clustering algorithm to overcome this bottleneck. This empirical study will adapt the existing approaches proven successful in other domains to the problem of clustering sequence data in IR-seq analysis.

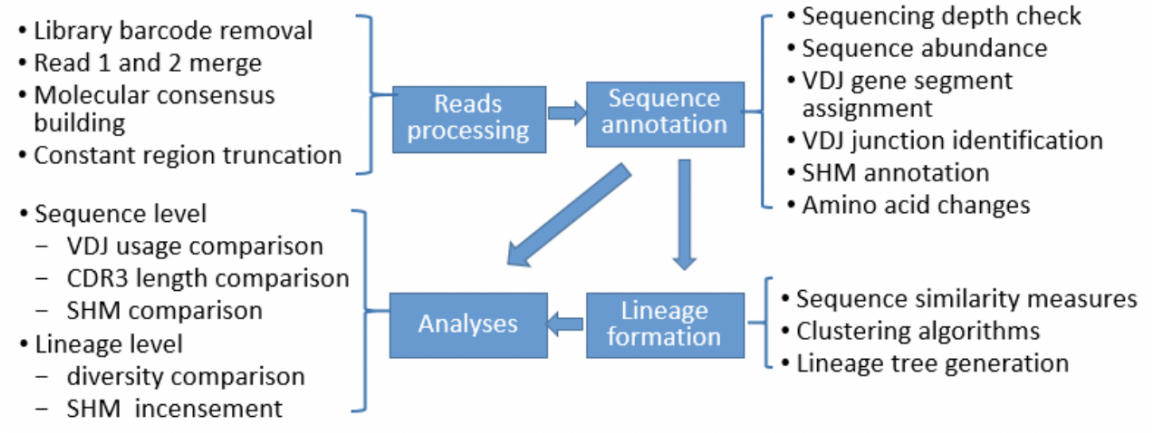

Figure 1.2: Overview of IR-Seq framework

1.2 Problem Definition

To determine the clones in IR-Seq dataset, sequences are first grouped based on identical V gene and J gene and junction length. Hierarchical clustering is performed for each group. The resulting hierarchy of each group is cut at a certain distance threshold (discussed later) to get the flat clusters which are called clones. The grouping is based on the assumption that the sequences belonging to the same clones are supposed to have the same length in the junction region and share the same V gene and J gene [12]. Note this assumption has excluded the case where mutations may change the junction length. In some extreme but rare case, the junction length may change, although the percentage is low around 1.3-6.5 % [30].

We tried to understand how the group size(V-J-Junction group) is distributed in IR-Seq dataset by using some of the real dataset used in our experiment. As shown in Figure 1.3 and Figure 1.4, for the four datasets(explained in detail later) , the group sizes are almost linearly related to the overall dataset sizes.X-axis represents the data size in thousand and the y-axis is the corresponding biggest group based on common V, J gene and junction length. With this projection, we expect the group size of around 25 thousand at two million of records in the dataset, and 140-180 thousand at 10 millions of records. The biggest group becomes even larger around 600-700 thousand for 10 million dataset when grouping

is done based only on the V and J gene (considering junction length change on mutation). Since clustering has to be performed on each group, and we may expect many such big groups, hierarchical clustering becomes the bottleneck for such big groups because of its $O(N^2 logN)$ computational complexity. So far, the most effective algorithm for finding the B-cell clones is single-linkage hierarchical clustering that generates the closest domain-specific clustering structures as shown by Gupta et al. [12]. Yet, its best complexity is $O(N^2)$ [23], which significantly restricts the scale of sequence datasets. Our goal is to investigate the methods that give (possibly approximate) single-linkage clustering structures with much faster processing time. For our experiment, we have used grouping based on common V, J, and Junction length because there is less chance(1.3-6.5 %)that mutation might change the junction length.

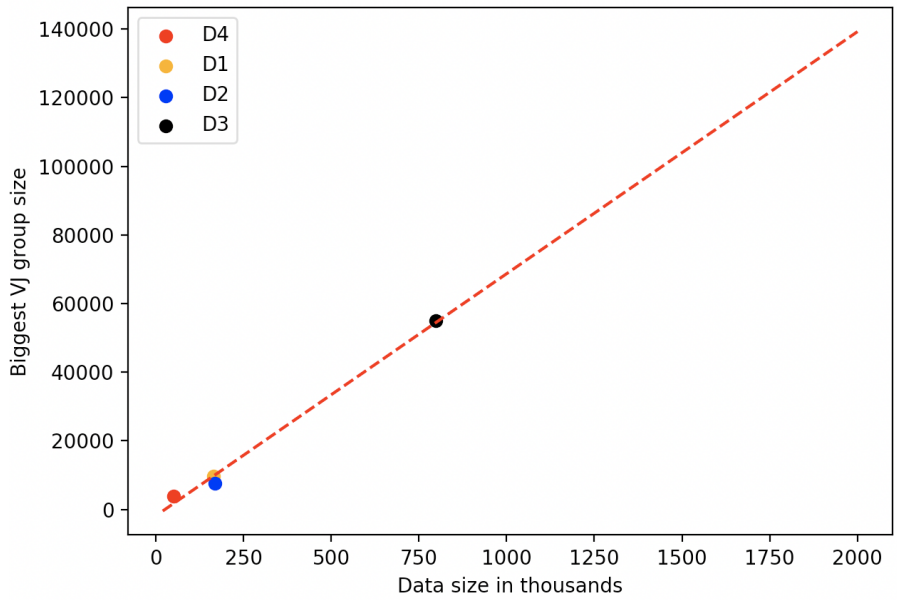

Figure 1.3: The largest size of group according to the overall dataset size when grouped based on V,J gene

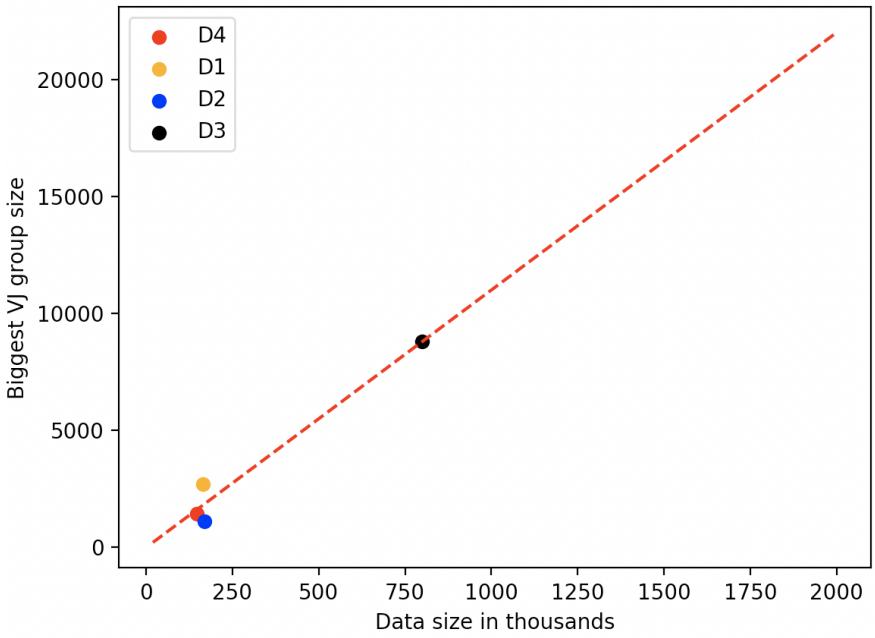

Figure 1.4: The largest size of group according to the overall dataset size when grouped based on V,J gene and Junction length

1.3 Strategies

One of our strategy includes processing all the V-J groups in parallel. This kind of parallel processing can be easily achieved using a spark. The major problem lies in clustering of these groups when the single group size is very large.So optimization is necessary for hierarchical clustering algorithm used to cluster such single big group. Here we discuss our three different strategies for fast processing of hierarchical clustering(HC) algorithm that works with sequence data.

First, we will experiment with non-Euclidean indexing methods customized for sequence data to reduce the complexity of key steps in the bottom-up agglomerative algorithms. Most existing agglomerative algorithms require the pairwise distance matrix as the input, which is not scalable. It is possible to avoid computing the entire distance matrix computation with an indexing structure for sequences, which can potentially reduce the complexity of the initialization stage from $O(N^2)$ to $O(N \log N)$. With a certain approxi-

mate method, the complexity of the intermediate iterative steps can also be reduced. Many such optimization methods have been based on the Euclidean space [24] and especially on low dimensional (2 or 3 dimensions) datasets for fast nearest neighbor search, as indexing only works effectively on low-dimensional Euclidean datasets due to the curse of dimensionality [15]. It's unknown how effective the sequence indexing methods are, which uses non-Euclidean metrics, such as edit distance, Hamming distance, and alignment-based distance [16, 11]. Our first strategy will be investigating the non-Euclidean indexing methods to speed up certain stages of hierarchical clustering and thus reduce the overall complexity.

Second, we also explore the tree-based summarization approach for sequence data. The idea is to maintain a height-balanced multi-way tree in a stream-processing style, which scans the sequence dataset once and absorbs each record into the summarization tree. BIRCH [26] has shown such a tree works for numeric data on Euclidean distance space. However, no study has shown that this idea also works for sequence data in non-Euclidean space and generates fast summarization results. We expect this summarization to be useful as preprocessing to reduce the input size for hierarchical clustering used in B-cell clones analysis.

Third, parallel processing can be effective for certain algorithm settings. There have been several efforts to develop the parallel or distributed version of the single-linkage algorithm. Olson [23] shows that using the parallel random access memory (PRAM) architecture, the overall complexity can be reduced to $O(N)$ with N processors for N sequences. Most recently, Jin et al. [14] have implemented the minimum-spanning-tree (MST) algorithm with Apache Spark. Since single-linkage clustering is equivalent to the MST problem, the SparkMST works as a parallel single-linkage algorithm. As Gupta et al. [12] has shown that single-linkage clustering is the best for B-cell colony analysis, we consider that the SparkMST as a promising candidate for large-scale parallel processing of sequence data. Note that the first two approaches also work for hierarchical clustering algorithms other than SLINK, while the parallel solution works only for SLINK so far.

1.4 Contribution

We summarize our unique contributions as follows.

- We have designed the framework for integrating cluster indexing structures for faster (and approximate) hierarchical agglomerative algorithm. It allows any non-Euclidean indexing method to plug in. We have experimented with the VP-Tree [29, 3] as the non-Euclidean indexing plugin. The result shows that this approach can preserve the clustering quality perfectly with good performance gain .

- We have developed the Sequence Condensation Tree (SCT) structure and algorithms for scanning the dataset once and absorbing sequences into a hierarchical clustering structure. It includes a novel design of node structure that can hierarchically and recursively summarize similar records under each tree branch. We show that this algorithm is highly scalable and resource efficient and good preprocessing approach for the index-based hierarchical clustering algorithm .

- We have implemented the Spark version of the MST-based clustering (SparkMST) [14] for processing sequence data. Our experimental evaluation shows that SparkMST scales well to both the size of sequence dataset and the size of the Spark cluster. It's a good candidate solution for users who can provision a large-scale Spark cluster and need only SLINK result.

- We have developed spark based parallel processing of all V, J groups for entire immune repertoire data analysis. Optimized indexed-enhanced HC, SCT based HC and MST based HC can be applied to the individual group to get the clustering structure within them. Our experimental result shows that all of our fast hierarchical clustering approaches greatly improves the computational cost for immune repertoire analysis.

- We have performed a comparative empirical study to understand the performance improvements of different approximate hierarchical Clustering(HC) approach.

1.5 Thesis Structure

The remaining sections of the thesis are organized as follows. In Chapter 2 we give information about Immune system , IR-Sequencing, the basic notations and definitions and some related work . Chapter 3 describes the detailed strategies for fast and scalable sequence clustering with a focus on optimizing the hierarchical clustering method. Chapter 4 describes some existing methods used for our comparative study and shows the experimental evaluation results. Finally, Chapter 5 concludes our work.

Background

2.1 The immune system

The human immune system is made up of different cells, tissues and organs working together to defend the body from foreign invaders. One of the important cells is white blood cells(e.g B-cell and T-cell). The foreign agent can be an infection causing organism such as virus, bacteria, fungi and so on. The human body provides the environment for them to enter and grow. So the immune system has to keep them out or destroy from our body. There are many body parts such as thymus, bone marrow, spleen, lymphoid tissue, lymph node and vessels which actively take parts in the immune system. Antibodies produced by B-cell are responsible to capture anti-genes which are destroyed by T-cells. Scientists are now able to mass produce immune cells and antibodies for detail study. This has not only revolutionized the study of the immune system but also has created a great impact in clinical study and research. Even though much research has been done to understand the immune system, there is plenty of opportunities to study how the human body destroys infected cells and how body shows immune response when vaccination has been applied

2.2 Immune Repertoire Sequencing(IR-Seq)

Immune repertoire sequencing (IR-seq) refers to the sequencing of both B cell immunoglobulin receptor (BCR) and T cell immunoglobulin receptor (TCR) and is considered useful in both research and clinical settings. A systematic study of IR-seq is important to understand the response of the adaptive immune system, to find new infectious agents, diagnosis disease and measure vaccination effects[18].

Using V(D)J recombination, the human immune system generates hundreds of millions of B T cells that each bear a unique antigen receptor gene. When B-cell recognizes the antigen it gets activated and divides into B-cell clones. Single original ancestral nave B cell could become several different sub-population.During division B-cell receptor undergoes a high rate of mutation which is due to single nucleotide substitution, insertion, and deletion. Organizing these cells into informatically defined lineages by examining their antibody gene sequences provided a means of study this important condensed evolution. T cell also follows a similar path to generate diversity however, they don't undergo somatic hyper-mutation(SHM) . This makes T cell IR-seq data easier to analyze compared to B cells.

Statistical properties of IR-seq data such as diversity, mutation, clone size distribution plays an important role in the identification of abnormalities caused by disease and aging. Accurately identifying clone size distribution can reveal many important information like species richness,minimal residual disease detection ,Shannon entropy and so on. Lineage analysis within the clone also has several applications. Once the sequence has been assigned to clonal lineage, the following information can be obtained:

- degree of diversification (species of new sequences and their abundance)
- amount of mutations from ancestor to progenies
- the antibody developmental path if samples from multiple time points are pooled together.

- degree of isotype switching (IgM to IgG or IgA)

IR-seq analysis is an interesting problem because there is a need to develop the method to find the sub-populations and there are a great many numbers of sub-populations in the repertoire. Different treatment for the disease (e.g. cancer) has been developed which affect the immune system, hence the size of repertoire keeps changing. So IR-seq analysis helps to monitor those effect by measuring the size of repertoire.

2.3 Sequence

A sequence in our study is a string of characters formed from the nucleobases, namely $\{$A,C,T,G$\}$. As the bases are typically paired chemically and sequences exist in pairs, the length of the sequence is also called the number of "base pairs". The typical sequence length for immune repertoire analysis is around 200-300, depending on the specific experimental design. In this paper, we will use lower cases, e.g., m or n to denote the sequence length, and N to represent the number of sequences.

2.4 Distance Measures

The similarity of sequences is defined with a certain measure. We list a few popular choices of the similarity measure.

2.4.1 K-mer based distance

K-mer refers to the substring of length k from the given string. In the field of bioinformatics, k-mer can be referred as all possible subsequence of length k from full-length DNA sequence read.All possible 3-mer for sequence 'ACTAGG' are ACT,CTA,TAG,AGG . So the k-mer distance between any two sequences can be defined as the number of unique

k mer that are shared between them. K-mer based distance can be used for approximate distance computation as this is computationally less expensive than other exact distance computation like edit-distance and sequence alignment.

2.4.2 Edit-Distance

Edit distance, also known as Levenshtein distance, is a standard metric for defining the dissimilarity between two strings by computing the minimum number of operations (insertion, removal, and substitution) required to transform one string to the other. Edit-Distance is a popular choice for strings of unequal length. However, it's quite expensive for sequences of 200-300 base pairs. For two sequences of lengths m and n, respectively, the overall complexity of edit distance is $O(mn)$.

Lets take two strings bitten, hitting .The edit-distance between these two words is 3 which can be explained by following operations.

- bitten $\rightarrow$ hitten (replace b with h)
- hitten $\rightarrow$ hittin (replace e with i)
- hittin $\rightarrow$hitting (adding g at the end)

Similarly consider two DNA sequences ACTAGAA, CCTAGT

- ACTAGAA$\rightarrow$CCTAGAA (substitute C for A at first position)
- CCTAGAA$\rightarrow$CCTAGTA (substitute T for A at 6th position)
- CCTAGTA$\rightarrow$ CCTAGT (delete A from Last position)

So the edit distance between two sequences ACTAGAA, CCTAGT is 3.

2.4.3 Sequence Alignment

Sequence alignment is the process of arranging the DNA or protein sequences so as to find the similarity between them. After the alignment, a small gap will be inserted between the nucleotide so that similar letters/nucleotides will be falling in the same column.

Before alignment After alignment

- ACATTG → ACATTG
- ACCG → AC- - CG

Dynamic programming is used to find the sequence alignment provided the particular scoring function (positive score if nucleotide match, negative score if mismatch or gap). After the alignment, the distance between sequences can be calculated as the number of mismatch in the nucleotide. Even though dynamic programming guarantees the optimal alignment, this method becomes very expensive when the length of the sequence is very large. The popular alignment algorithms, such as Smith-Waterman [25], are quite expensive.

2.4.4 Hamming Distance

Hamming distance is measured between two equal length string. It is defined as the number of position at which corresponding characters differ. As the typical immune-related mutations happen as base substitutions and thus the length of sequence keeps unchanged, Hamming distance is a valid choice for B-cell colony analysis [12].

2.5 Related Work

Many well-known fast sequence clustering algorithms, such as CD-HIT [16], UCLUST [10], DNACLUST [11], are based on simple threshold-based grouping, which does not

give the ideal clustering structures. CD-HIT[16] first sorts sequences by length ,so that the longest sequence is selected as the first seed cluster. Each of the remaining sequences is compared with the existing seed clusters, and either merged to the seed clusters if the distance to the seed is less than a predefined threshold or becomes a new seed cluster if not existing ones can absorb it. It also uses k-mer based filtering for fast processing. The algorithm has the complexity of O(Nn) where $n < N$, n is the total seed sequences, and N is the total number of sequences. With tight thresholds, n is often very large, e.g., up to thousands, leading to slow processing. The core idea of UCLUST[10] is similar to CD-HIT[16] and has similar complexity. It uses a fast sequence search heuristic to find the closest seed sequence, which helps when n grows large. DNACLUST[11] is another similar threshold-based approach, but using edit-distance as the similarity measure. Apparently, this category of methods only work as rough grouping methods to aggregate very similar ones (e.g., with a threshold $> 95\%$ of sequence length). With a relaxed threshold, the algorithms do not give ideal clusters and the clusters highly depend on the order of processed sequences.

Dendrograms generated from hierarchical clustering have been one of the most intuitive methods for clustering analysis in biomedical research. However, the general hierarchical clustering algorithms optimized with heap have the complexity of $O(N^2 \log N)$, which does not scale well for large sequence datasets. Biomedical researchers have been using the expensive algorithms and enduring the long running times, until the size of data grows to an acceptable level. Recently, Cai et al. propose the ESPRIT-Tree [4] method, aiming to address the performance issues with the million-level sequence set. It uses k-mer and Alignment based distance computation and a leveled tree to organize sequences. Each layer in the tree has a specific layer-based distance threshold: the distances between nodes are all larger than the threshold, while records covered by the node have distances to the node center less than the threshold. The tree is used for nearest neighbor search for a hierarchical clustering algorithm. It worked well on the microbial RNA sequences. However, our experiment shows that it does not give the top-quality clustering results and its C++

implementation is constantly slower than our Scala-based VPT-HC.

The general agglomerative hierarchical clustering framework does not have good inherent parallelism, as each decision of cluster merging has to depend on previous merging results. HPC-CLUST [21] tried multi-threading approach to achieve parallelization in the distance calculation step, which does not address the scalability bottleneck. Swarm [19] uses a two-phase agglomerative single linkage algorithm and tries to address the input order dependency. However, the algorithm is quite slow and does not scale well to large data. For the specific type of hierarchical algorithms: the single-linkage algorithm, Olson [23] has shown that $O(N)$ complexity can be achieved with $O(N)$ nodes under a shared memory PRAM architecture or with a $O(N/\log N)$-node butterfly network. However, such special architectures are not accessible to most biomedical researchers. Recently, Jin et al. [14] propose a parallel MST-based graph clustering algorithm, equivalent to single-linkage clustering that can be implemented with a Spark cluster on commodity servers. We have investigated its scalability on sequence datasets and found good performance with the increased data size.

The tree-based summarization methods have been used in stream clustering, e.g., BIRCH [26] and CluStream [1] for multidimensional vectors in Euclidean space, and HE-Tree [6] for categorical data. To our knowledge, no work is reported on sequence data. We design the SCT approach for sequence data and shows that it has very low processing costs and is highly scalable for large datasets. Post-processing algorithms can be developed to work with a much less number of summarized nodes to generate clustering results close to the single linkage's or any other hierarchical clustering.

Approximation methods to speed up Hierarchical Clustering(HC) for sequence data

Since hierarchical clustering gives the best results for B-cell clones analysis, our strategies for developing scalable clustering algorithm will be around optimizing this algorithm. First, we will keep the agglomerative framework and optimize its key steps with indexing structures. The progressively merged subclusters will be organized with a non-Euclidean indexing structure. This indexing structure is basically used to support fast nearest neighbor search. The index-enhanced framework will be compared with a recently developed ESPRIT-tree [4] for 16sRNA Sequence-based microbial community analysis. Second, we develop the SCT tree-based summarization approach for processing the large sequence data in one scan, which will be highly scalable even with only a single machine. We will evaluate how the generated summarization can be useful as preprocessing for index-enhanced hierarchical clustering. Finally, we still keep the single-linkage algorithm unchanged, while evaluating its parallel implementation based on Jin et al. [14] Spark MST algorithm. Jin et al. have experimented the Spark-based implementation for vector data up to two millions of records. However, There is no study yet about its scalability on IR sequence data. We adopt the Spark MST algorithm and extend it to sequence data.

In the following, we will give more details of the three strategies and also conduct theoretical analysis on the potential performance gain.

3.1 Speeding up Hierarchical Clustering(HC) with Indexing

Consider the general sequence-based agglomerative clustering framework (Algorithm 1). There are two steps: Step 2 and Step 8 involving the nearest neighbor search.

Algorithm 1 Sequence Agglomerative Single-Linkage Clustering (S, t, δ)

1: **input:** sequence set S, the distance threshold t for stopping merge, and the distance function $\delta(s_i, s_j)$ for any pair of sequences.
2: consider each sequence $s_i \in S$ as a cluster and find its nearest neighbor s_i.NN, which results in a tuple (s_i, s_i.NN, d_i)
3: initialize a priority queue q with the tuples (s_i, s_i.NN, d_i), sorted by d_i.
4: **while** the merged cluster has distance $< t$ or the number of clusters > 2 **do**
5: (s, s.NN, d) $\leftarrow q.dequeue$;
6: s' $\leftarrow$ merge s and s.NN;
7: update the tuples in q, whose nearest neighbor is s or s.NN;
8: s'.NN $\leftarrow$ find the nearest neighbor of s' that has the distance d';
9: insert (s', s'.NN, d') into q;
10: **end while**
11: return the remaining clusters in q.

Therefore, we consider using a sequence-based indexing tree to speed up the nearest neighbor search steps. There are two key requirements on the desired indexing structure (1) it works on non-Euclidean metric space, and (2) it indices clusters of sequences, not individual sequences. For the first requirement, we consider adopting existing indexing methods for general metric spaces, such as the Vantage-Point tree (VP tree) [29, 3], and Cover Tree [2]. For the second requirement, we use a representative-sequence method to approximately sketch the boundary of a cluster. For single-linkage clustering, the distance is defined as the minimum pairwise distance between two sets of representative points from

the two compared clusters, respectively. We describe the key algorithm for maintaining the representative sequences when merging two clusters.

Maintaining Representative Sequences in Merging. In the above framework, the indexing algorithm will build an index on cluster objects, the distances between which have to be defined. The cluster objects start from one sequence and can grow to large size after the merging operations. For the single-linkage algorithm, the cluster distances can be updated incrementally: when merging a pair of clusters s and its nearest neighbor s.NN, the nearest neighbor of s.NN can be used as the nearest neighbor of the merged cluster, when s and s.NN are not mutual nearest neighbors: i.e., s.NN.nearest != s. When they are mutual nearest neighbors, the index on clusters is used to find the nearest neighbor of the merged cluster.

We use representative sequences in each cluster for approximate cluster-cluster distance computation. Traditionally, the cluster distance in hierarchical clustering is defined based on the pairwise distances between pairs of members from the two clusters, respectively, e.g., the minimum, average, and complete linkage definitions, which are quite expensive. We try to extract the representative sequences likely around the exterior boundary of the cluster by the following recursive method. (1) If the cluster contains only one sequence, that sequence becomes a representative sequence and the center sequence, automatically. (2) Assume two clusters C_1 and C_2 are merged, the representative sequence sets of which are R_1 and R_2, respectively. Let set sizes be $|R_1| = n_1$ and $|R_2| = n_2$. If $n_1 + n_2 > k$, where k is the pre-defined maximum number of representative points per cluster, compute the total sum of squared distances, τ_i, for each candidate sequence r_i in the union set $R_1 \cup R_2$.

$$\tau_i = \sum_{j \neq i} distance^2(r_i, r_j)$$

This total sum of squared distances for r_i can approximately capture the location of the sequence inside the cluster: the ones with the largest sums are around the boundary of the

cluster. Thus, we can keep the top k candidates that have the largest total sums as the representative sequences in the merged cluster. Similarly, we also keep track of the *center sequence* of the cluster that has the smallest total sum. The overall update cost is controlled by the parameter k.

Cost Analysis. Ideally, the index building time is about $O(N \log N)$ and the nearest neighbor search costs $O(\log N)$ distance computations. Thus, the cost of the initial step of finding each sequence's nearest neighbor is reduced to $O(N \log N)$ distance computations. The steps in the loop involve one nearest neighbor search and possibly $O(N)$ comparisons with the representative-sequence based cluster representation. Thus, with the help of indexing, we can reduce the number of expensive distance computations to $O(N \log N)$. We will evaluate the costs in Section 4 with the VP-tree as the indexing structure, and compare with a recently developed index-enhanced sequence clustering method ESPRIT-tree [4].

3.2 Sequence Condensation Tree (SCT) - a fast sequence summarization algorithm

To study more efficient single-machine processing methods, we develop the SCT fast sequence summarization algorithm that scans the dataset for only once and sketches the summaries in the dataset. The SCT tree is a multi-way balanced tree. It grows from a single root node to multiple layers with the inserted sequence. The SCT sequence insertion algorithm quickly routes a new sequence along the right path on the SCT tree to the expected node that absorbs the sequence, via a series of node-sequence similarity computation. Thus, the cost of processing one sequence is fast, $O(\log N)$ at most.

The algorithm starts with one empty root node. The tree grows incrementally with inserted sequences. When a new sequence comes, it follows the root node down to the leaf by comparing the similarity between the sequence and the node. Each time the child node

with the highest similarity (or lowest distance) is selected to move down until it reaches the leaf. The leaf entries are checked to find the similar one (within a threshold t, distance(leaf entry, sequence) $< t$), to absorb the sequence. If no leaf entry can absorb, a new entry is created to contain that record. The creation of new entry may cause node split when the pre-defined maximum number of entries is reached. Node split may be propagated up to make the tree balanced, a similar process used by multi-way balanced trees. The core similarity computation is defined between node and sequence. To describe the cluster defined by the node, we have used the representative-sequence method defined previously in Section 3.1. While a new sequence is inserted into the node, we use the total sum method to check whether this sequence can be a new representative sequence for that node. Based on the representative sequences, we have defined the following methods for node-sequence similarity computation.

- **Center based:** A center record is dynamically maintained with the previously described method. The center-based node-sequence similarity is simply defined as the distance between the center and the sequence.

- **Average/Min:** Alternatively, we consider using the average or minimum distance among all pairs of representative sequence and the inserted sequence to define the node-sequence distance.

A simple structure of SCT is illustrated in Figure 3.1. The SCT has 3 levels with leaf nodes at Level 0, the root node at Level 2, and intermediate/non-leaf nodes at Level 1. Sequences at leaf nodes are absorbed into a leaf entry according to a predefined threshold t. Typically, in B-cell colony analysis, we don't need to separate very similar sequences, e.g., those having their similarity higher than 95% of the sequence length, and they are merged into one leaf entry.

The SCT has several unique advantages for clustering sequence data.

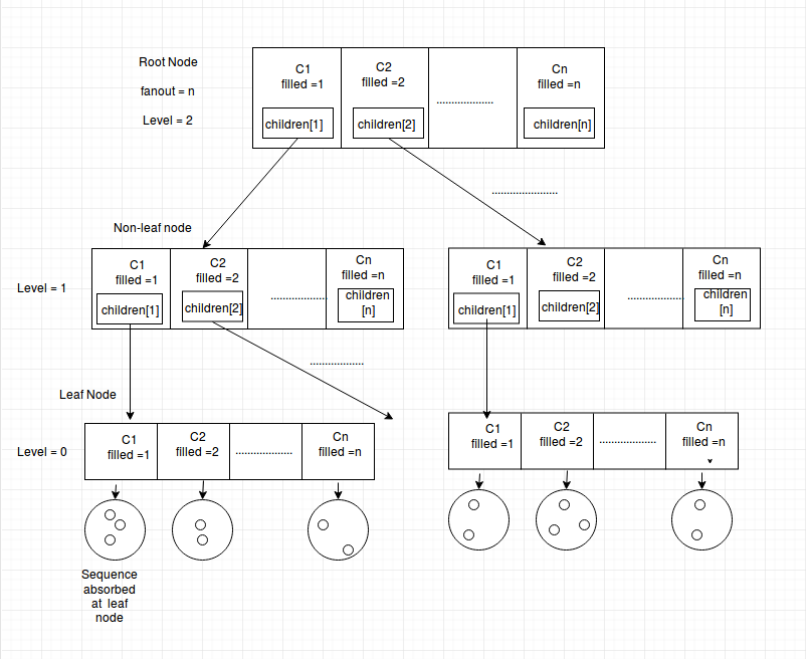

Figure 3.1: Structure of Sequence Condensation Tree (SCT)

- The whole sequence dataset is scanned only once and thus scalable to large sequence sets.
- The cost of processing one sequence is determined by the height of the tree, which is approximately $O(\log_f N)$ for N sequences, where f is the number of entries per node. For large f, e.g., f=10 or 20, the height is very limited.
- The tree can serve as a preprocessing tool for quickly filtering out singletons and characterizing major groups, e.g., the size, scale (defined by the representative sequences) and the center).
- Thresholds can be applied to select the nodes for fast post-processing, e.g., generating hierarchical clustering structures based on the selected nodes.

When using SCT as the preprocessing step of HC, we choose a threshold, T , between the leaf-level threshold and the bimodal threshold to identify the summary nodes in the tree those nodes have their diameters $< T$, while their parents have diameters $\geq T$. Figure 3.2 shows simple diagrammatic representation how the summary node can be obtained while

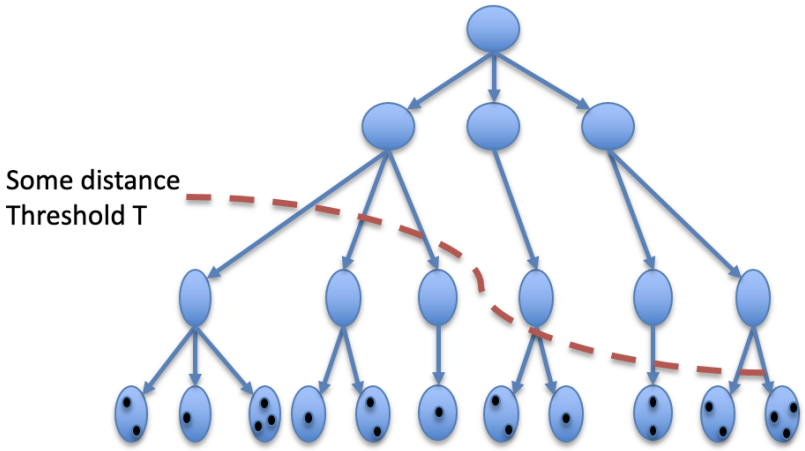

Figure 3.2: Cutting of Sequence Condensation Tree (SCT) at threshold T

cutting SCT at threshold T. These summary nodes are used as the input to the standard HC algorithm to find the final clusters. Our experiments have shown SCT can generate a significantly smaller number of summary nodes than the number of sequences in the original dataset, which significantly reduces the overall complexity of HC.

Experiments

The goals of the experimental evaluation include: (1) the scalability approximate hierarchical clustering : the index-enhanced hierarchical clustering approach and the SCT summarization tree approach, (2) the scalability of multi-server Spark-based method: SparkMST, (3) the clustering quality of the approximation methods, including the index-enhanced approach and SCT, (4) time cost comparison of clustering entire repertoire (all v,j group) for different algorithms.

First, let us discuss some of the existing HC methods used for comparative analysis with our approximate methods.

4.1 Existing methods used for our comparative study

4.1.1 Parallel Single Linkage Clustering

Another possible option is to scale up the processing with a parallel processing computing cluster, e.g., the Spark [31] based approach. The hope is to develop an algorithm that speeds up nicely with the increased size of the Spark cluster. There have been several efforts to design a parallel or distributed version of single linkage hierarchical clustering algorithm in the past [23]. However, they work on the traditional shared memory systems, not the cheap commodity servers or virtual machines, available in the public clouds for most users. Recently, Jin et al. [14] have shown that single linkage hierarchical clustering can be

possibly implemented using Spark. The core idea is to find the single-linkage structure with the minimum spanning tree (MST) method, which is proved equivalent to single-linkage clustering.

The MST method can be possibly parallelized in two stages. In the first stage, the whole dataset is randomly partitioned into p smaller subsets. The complete graph is formed with each small subset, where nodes are the records, and edge weights are the distances between the connected records. In total there are p complete graphs and $O(p^2)$ bipartite graphs. To avoid computing all edge weights the Prim's algorithm is used for finding the MST of each of these graphs. In the second stage, the MSTs are merged with the Kruskal's algorithm that works only the extracted edges. Figure 4.1 describes the parallel version of the single linkage clustering.

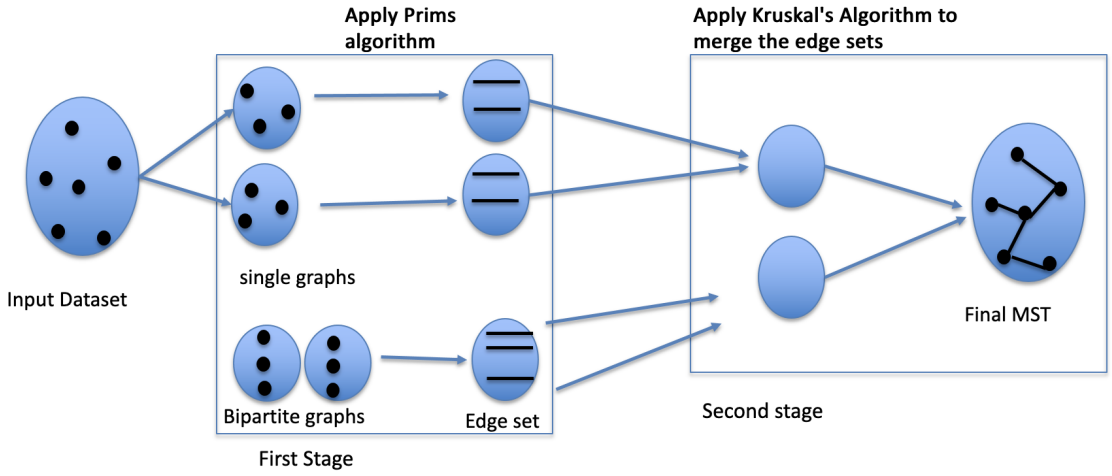

Figure 4.1: Parallel Single Linkage Hierarchical Clustering approach based on MST.

Cost and Scalability Analysis. The complexity of the Prim's algorithm is $O(N^2 \log N)$ with the help of a priority queue for N sequences. With p partitions, both the complete graph and the bipartite graph MST algorithms have the same complexity $O((N/p)^2 \log(N/p))$. Assume there are w parallel workers in the Spark cluster, we will need to schedule $O((p^2 + p)/w)$ rounds to process all the graphs. Thus, the overall parallel running time for the first stage is $O((p^2+p)/w(N/p)^2 \log(N/p)) = O(N^2/w \log(N/p))$, which ideally will linearly

scale with the number of parallel workers, w. The output of the Prim's algorithm will contain $O(N/p)$ edges for both types of graph. Thus, there are $O(p^2)$ of such edge sets. With the Kruskal's algorithm, we can progressively merge pairs of edge sets in a hierarchical way in the second stage. There are $O(p^2)$ merges, each of which costs $O((N/p)\log(N/p))$. With w workers, the second stage costs about $O(Np/w\log(N/p))$, clearly less expensive than the first stage. Overall, the algorithm should scale nicely to the number of working nodes w.

We have revised the SparkMST algorithm to work with the sequence data. As the MST algorithm generates the same result of single-linkage clustering, the quality of clustering is fully preserved compared to other methods we have studied. The experimental evaluation also shows its good scalability. While with a small number of computing nodes, the performance gain might be slight, a larger Spark cluster does help reduce the cost.

4.1.2 ESPRIT-Tree

ESPRIT-Tree is approximate hierarchical clustering algorithm described by [4] , originally used for microbial community analysis. This method cluster the sequences in two stage. First partitioning tree is constructed by considering different threshold at each level i.e parent node has a higher threshold than the child node and parent node radius covers the child nodes radius. Then clustering is done recursively by finding the Nearest Neighbor(NN) from the tree and merging the closest cluster. We have obtained the implementation of ESPRIT-tree from authors website and modified the output according to our need for further analysis and compared it with our index-enhanced hierarchical clustering algorithm.

4.2 Setup

We have implemented all the candidate algorithms for sequence data. VP-tree is used as the indexing method for the index-enhanced hierarchical clustering approach(VPT-HC). We compare this approach with a somewhat similar approach ESPRIT-Tree [4] that was developed for clustering microbial RNA sequences and analyzing the microbial communities. The ESPRIT-Tree implementation is available as a compiled binary from C++ source code. All our algorithms are implemented with Scala.

The single-machine algorithms are evaluated on a Linux server equipped with Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz,16 core ,250GB memory. For parallel processing, we have used a standalone cluster that has 11 powerful nodes (1 head node + 10 slave nodes), each of which has 16 cores and 250 GB memory. This enables us to set up a Spark cluster with up to 160 parallel workers.

4.3 Quality Measures and Baseline Establishment

Since the recent study has shown that the single-linkage algorithm is the best option for B-cell colony analysis, we have adopted the standard single-linkage results as the baseline for quality evaluation. However, most standard packages (e.g., in R or Python) are not able to handle the scale of our sequence data, e.g., requiring the whole distance matrix as the input. We have used the SparkMST algorithm to generate the baseline.

The ground truth cluster labels are generated based on the SparkMST clustering results. It is done by estimated ideal distance cut, based on a global sample-distance based threshold identified using distance-to-nearest plot. With the baseline labels, we have used Normalized Mutual Information(NMI) for determining the quality of a clustering result, compared to the baseline clustering labels. Let the ground truth cluster labels be the ”true labels” drawn from a random variable T, and the clustering algorithm assignments are

the "predicted labels" drawn from another random variable P. Let H(T) and H(P) be the entropy of the label distributions, respectively, and $I(T;P)$ be the mutual information between the two random variables. NMI(T; P) is defined as

$$NMI(T;P) = \frac{2I(T;P)}{H(T) + H(P)}$$

To determine the cluster labels, hierarchical clustering is cut at a fixed distance threshold. The threshold is determined using the following method. Gupta et al. and Ning et al. [12, 13] have shown that B-cell mutations follow a bimodal distribution over the distance-to-nearest graph, as shown in Figure 4.2. We find the nearest neighbor for each junction sequence in a sample dataset and then plot the distribution (e.g., the histogram) of the distances to their nearest neighbors. The ideal distance cut is approximately located at the valleys between the two modes. The intuition behind this method is that the closely related sequences forming clones are on the first mode and the remaining (sparse) sequences are on the second mode. Our experiments on various real datasets also support this bimodal distribution. We use this method to generate the cluster labels in clustering quality evaluation.

4.4 Datasets

We have evaluated the algorithms with the following datasets:

- Malaria Dataset: This dataset is used in studying the B-cell clones of Malaria patients [27]. The blood samples are from one patient at four-time steps: first-year-pre-malaria, first-year-acute-malaria, second-year-pre-malaria, and second-year-acute-malaria. We have two datasets under this category. The first one let's say D1 which consists of around 164 thousand sequences, having different V, J combination. And

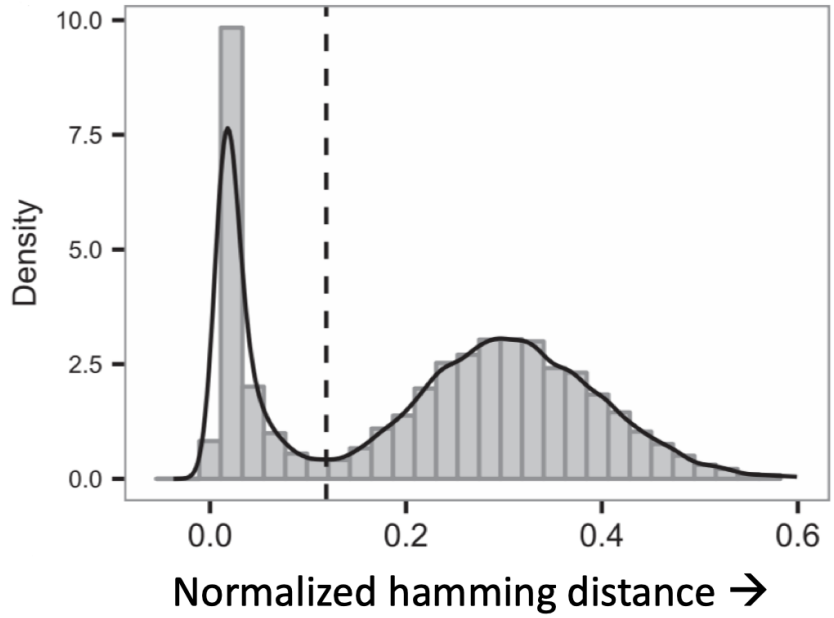

Figure 4.2: Threshold determination by distance-to- nearest-neighbor method (Credit [12]).

the other one lets say D2 which consists of around 169 thousand sequences, having different V, J combination.

- The real dataset shared by Gupta et al. [12]. It Consists of two different datasets.Raw data is available on SRA under BioProject accession no. PRJNA349143 and PRJNA338795 .Processed data used for our analysis are directly obtained from the authors. We consider them as D3 and D4 for our representation.D3 consist of sequences collected from blood samples of a healthy adult as part of an influenza vaccination study(800k records).D4 consist of sequence collected from blood sample used for the study of myasthenia gravis(250k records with Healthy Doner around 50k). Processed data is in a tabulated format with V(D)J assignment on raw data.

4.5 Result and Discussion

Clustering Quality. We first show the clustering quality of the different methods based on the NMI measure. To determine the cluster labels we first identified the threshold to cut the hierarchical clustering algorithm which is described in Gupta et al. [12].We basically generated a bimodal distance-to-nearest graph for each dataset and identified threshold as the minimum valley between two peaks. We considered the top 10 biggest groups having common V, J, and Junction length and identified true labels using MST based single linkage algorithm(sparkMST) by cutting the dendrogram at the previously identified bimodal threshold. We used the same threshold to cut the dendrogram for VP-Tree enhanced hierarchical clustering algorithm denotated as 'VPT-HC' and produced labels for calculating NMI.

Parameter Setting for SCT: We wanted to find the optimal parameter setting for the SCT method because cutting the tree at different threshold gives a different summary size. Threshold T for the summary node (previously discussed in section 4) between bimodal threshold and leaf level threshold is considered. We look at the rate of reduction of original data size to summaries by SCT based on different threshold cut. We plotted the different range of threshold T on x-axis and rate of reduction at the corresponding threshold on the y-axis as shown in Figure 4.3. Reduction rate is calculated as Rate %= (1 - SCT summaries/original data size) * 100 . We choose the threshold cut which has the highest reduction rate. We want to make sure that this threshold cut has no significance drop in quality as shown in Figure 4.4. We found out from Figure 4.3 and 4.4 that cutting the SCT tree closely at the bimodal threshold and applying VPT-HC gives us good result with a significant reduction on cluster size with accuracy intact. We have used this analysis to calculate the accuracy of SCT-VPT-HC(summary node from SCT used as input to VPT-HC) algorithm and presented the result in Table 4.1.

We computed the mean NMI and Standard deviation and presented the result on Table 4.1 for different Datasets. We used the same top 10 groups for each dataset and identified

Method	Dataset D1	Dataset D2	Dataset D3	Dataset D4
VPT-HC	0.95 ±0.07	0.99± 0.00	0.97 ±0.06	0.99± 0.00
SCT-VPT-HC	0.93 ±0.07	0.98± 0.02	0.96 ±0.06	0.99± 0.00
ESPRIT-Tree	0.77 ± 0.29	0.98± 0.02	0.92 ±0.07	0.95± 0.01

Table 4.1: NMI scores on Different Dataset. $0 <= \text{NMI} <= 1$. The larger, the better.

labels using ESPRIT-Tree for previously identified bimodal threshold.

From the Table 4.1 it is clear that VP-Tree enhanced algorithm (VPT-HC) has the highest NMI score. For Dataset D2 and Dataset D4 the quality of VPT-HC is almost perfect. The quality of SCT-VPT-HC is very close to VPT-HC for all the datasets. The ESPRIT-Tree has quality significanlty lower than VPT-HC and SCT-VPT-HC for all the Datasets.

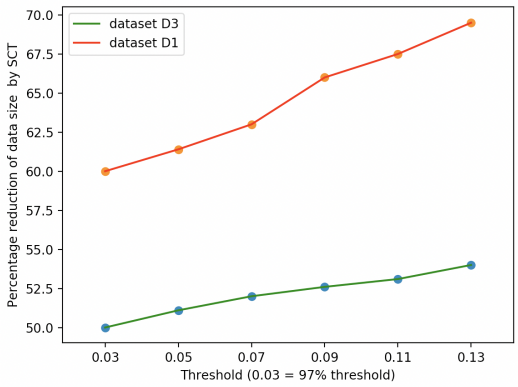

Figure 4.3: Data size reduction rate with different threshold(T) using SCT

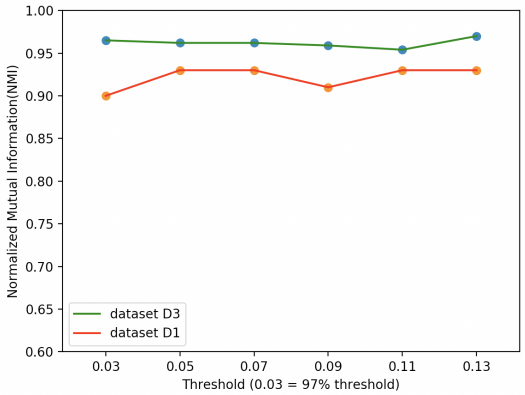

Figure 4.4: NMI value after dataset size is reduced by SCT and VPT-HC is applied on reduced groups

Scalability of approximate hierarchical clustering algorithms. To evaluate the computational costs of approximate hierarchical clustering algorithms and their scalability, we sampled up to 20 thousand junction sequence having the same length from the dataset D3. In Figure 4.5, the x-axis represents the sequence size in thousands and y-axis is the time (seconds) costs. The result shows that VPT-HC and ESPRIT-Tree have a similar pattern, slightly higher than linear complexity, which matches our analysis of complex-

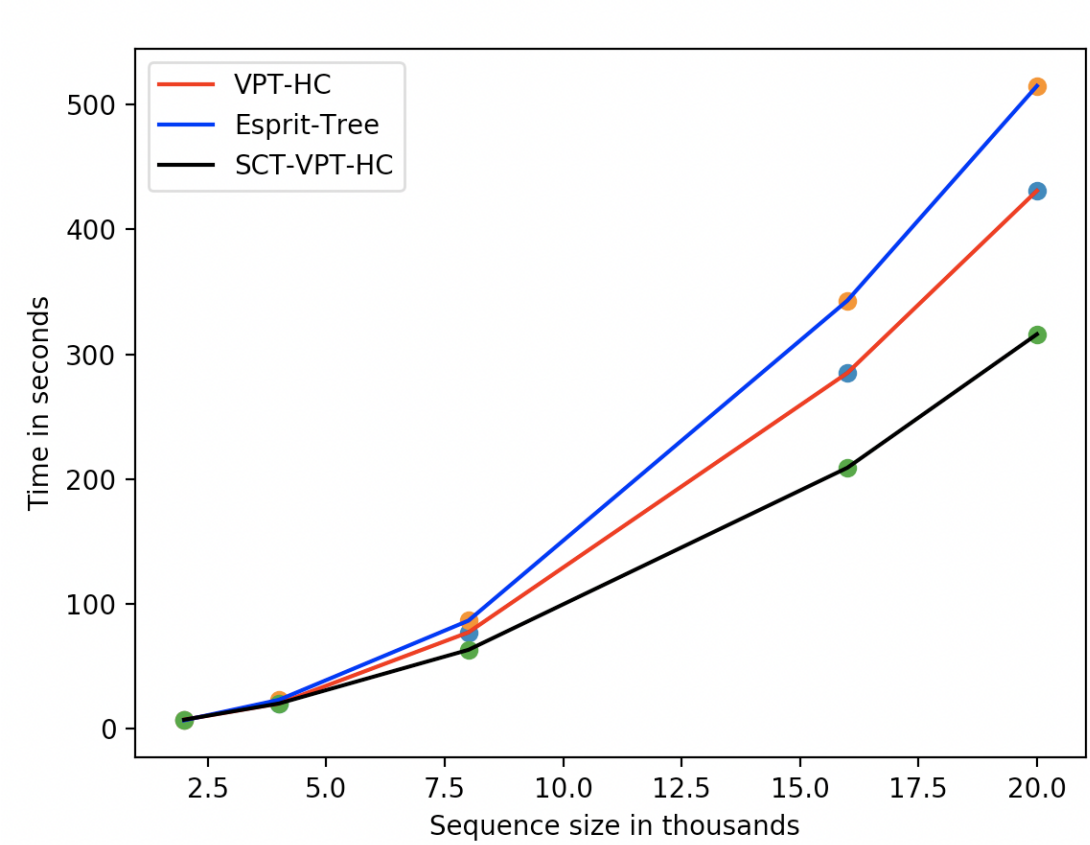

Figure 4.5: Cost comparison between different methods on Junction Sequence sampled from Dataset D3, for single-machine algorithms

ity $O(N \log N)$. SCT-VPT-HC is fastest among all the method because SCT significantly reduces the original input data to summaries.

Scalability of Spark based hierarchical clustering method:. We have performed another experiment to study the scalability of spark based single-linkage hierarchical algorithm. We study the scalability based on and how it scales with the increasing number of computing nodes for the same size of data. For this, we changed the Spark cluster configuration with 2, 4, 6, 8, and 10 worker nodes, respectively.

We plotted two figures to understand the scalability of Spark based algorithm when computing nodes are increased. Figure 4.6 shows the speedup when the computing nodes are increased from 2 to 10. The speedup is calculated by considering 2 computing nodes as a minimum number. We calculate speedup as $(2 * t2)/tn$ where t2 is computation cost

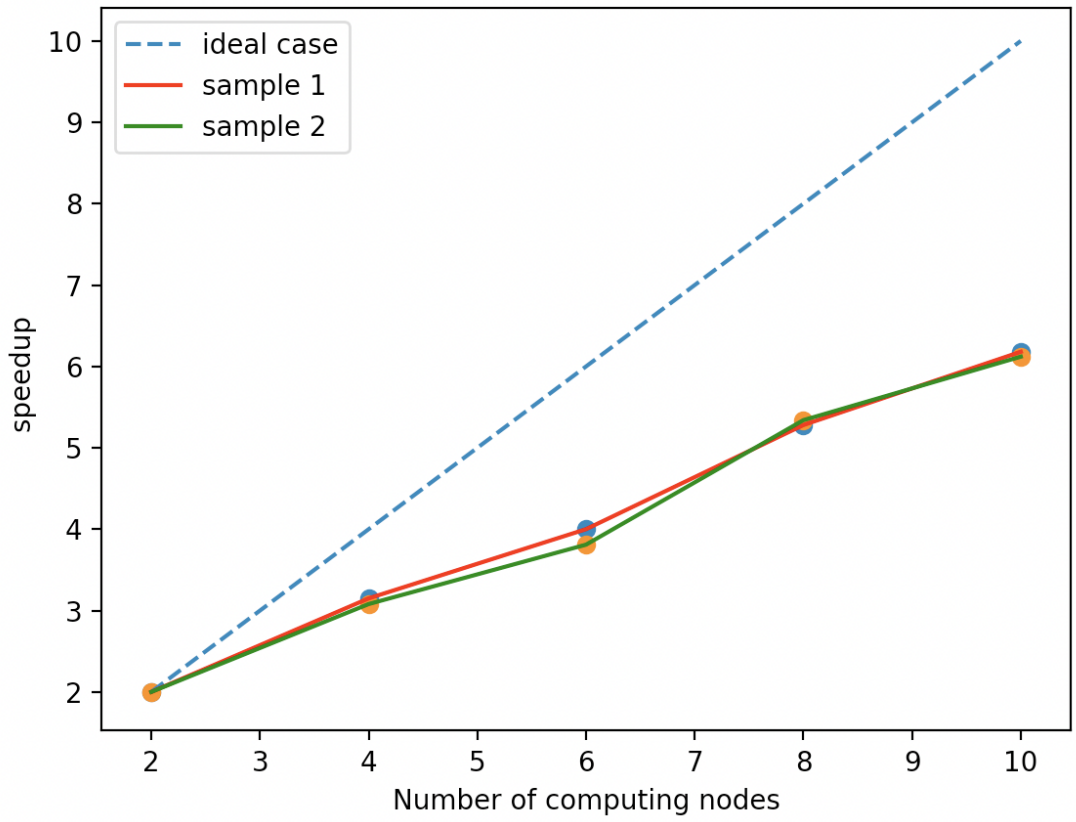

Figure 4.6: Speedup when the computing node are increased

while using 2 nodes and tn is the computation cost while using n nodes. From Figure 4.6 we see that good speedup is achieved with spark based method, slightly lower than the ideal case.

Figure 4.7 also shows a good trend and significant time gain when computing nodes are increased.

Cost comparison of approximate HC and spark based algorithm for fixed data size: We performed the experiment to understand the time cost of our two methods VPT-HC and SCT-VPT-HC with some existing approach like ESPRIT-Tree , Python implementation(SciPy).We also compare them with SparkMST. For this experiment, we keep the fixed data size 20k sequences and recorded the time for execution of all these methods. From Table 4.2 it is clear that there is a significant improvement of time by VPT-HC and SCT-VPT-HC and they work for general HC framework. SparkMST with 11 node cluster

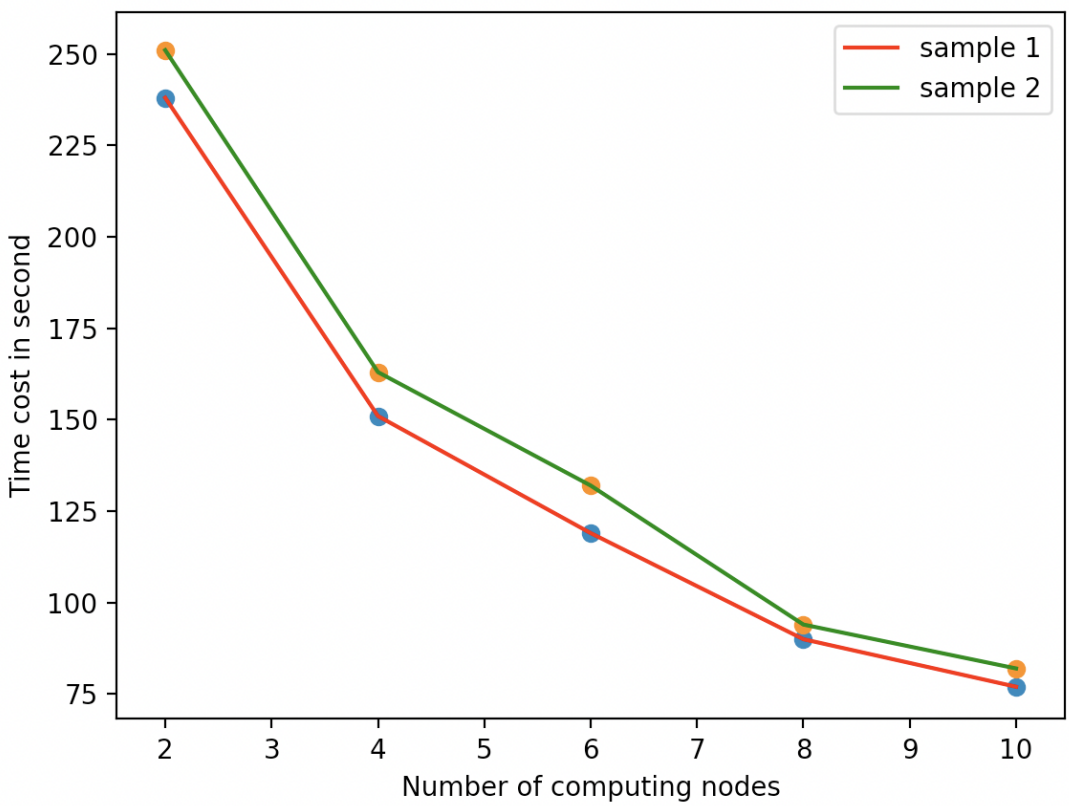

Figure 4.7: Cost for computing Spark Based clustering when computing nodes are increased(60k sequences)

is fastest among them but it works only for single linkage.

Cost comparison for processing entire repertoire (all v,j group): We performed the experiment to understand the cost of computing clustering for entire repertoire dataset. It involves grouping the sequence based on common v,j gene, and junction length and performing hierarchical clustering on each group. For this, we have developed spark based parallel grouping and applying HC on each group. Cost comparison of the different algorithm for all our real dataset is shown in Figure 4.8. For this evaluation, we have used VPT-HC,SCT-VPT-HC and LocalMST for processing individual group. LocalMST here represents the single linkage hierarchical clustering using MST without spark.

Result form Figure 4.8 shows that there is significant time improvement with SCT-

Method	Time(sec)
SciPy	1200
ESPRIT-Tree	515
VPT-HC	431
SCT-VPT-HC	316
SparkMST	35

Table 4.2: Time cost comparison for different algorithms for fixed data size.

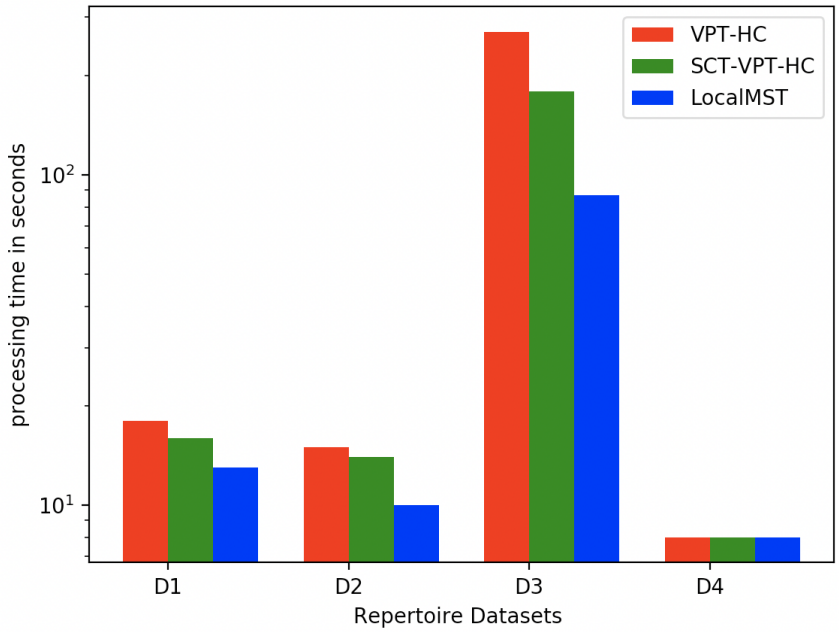

Figure 4.8: Cost comparision for entire repertiore analysis (time in sec).

VPT-HC from VPT-HC. MST based method is fastest among all.

Discussion. Based on the evaluation results, we can conclude that all the approaches work with different levels of performance gain. For the approximate algorithms, the index-enhanced methods can effectively bring down the cost of the original hierarchical clustering(HC). We will experiment with other non-Euclidean indexing structures, such as Cover Tree [2], as the plugin to the framework. In comparison, the SCT method is very fast which quickly summarizes the data and those summaries can be used with index-enhanced hierarchical clustering framework. Finally, SparkMST gives the exact single-linkage clustering results. SparkMST algorithm scales well for an increasing number of computing nodes.

Conclusion

Recent studies have shown that hierarchical clustering is the best method for B-cell clones analysis.However, the scale of sequence data for B-cell clones analysis is rapidly increasing, to a level that the classical hierarchical clustering algorithm cannot handle anymore. In this thesis, we study two strategies to scale up the hierarchical clustering algorithm for large sequence datasets: the index-enhanced hierarchical clustering, the SCT single-scan fast sequence summarization algorithm .And we also performed comparative analysis of our developed methods with parallel hierarchical algorithm SparkMST. We have done an extensive experimental evaluation on four real datasets. The result shows that the index-enhanced hierarchical clustering can preserve the clustering quality almost perfectly, SCT method can be very good for fast summarization and both these algorithms are memory efficient. We have implemented SparkMST parallel algorithm and found that SparkMST can scale well when the number of computing machines is increased but SparkMST is applicable only with single linkage. All these scalable hierarchical clustering algorithms show great performance improvement for entire immune repertoire sequence data used for B-cell clones analysis.

Bibliography

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *International Conference on Very Large Databases*, pages 81–92, 2003.

[2] Alina Beygelzimer, Sham Kakade, and John Langford. Cover tree for nearest neighbor calculations, 2006.

[3] Tolga Bozkaya and Zehra Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 24:361–404, 09 1999.

[4] Yunpeng Cai and Yijun Sun. Esprit-tree: hierarchical clustering analysis of millions of 16s rrna pyrosequences in quasilinear computational time. *Nucleic Acids Research*, 39(14), 2011.

[5] Keke Chen, V. S. A. Gogu, Di Wu, and Jiang Ning. Colt: Constrained lineage tree generation from sequence data. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 102–106, 2016.

[6] Keke Chen and Ling Liu. HE-Tree: a framework for detecting changes in clustering structure for categorical data streams. 18, 2009.

[7] Scott D Boyd, Eleanor L Marshall, Jason D Merker, Jay M Maniar, Lyndon N Zhang, Bita Sahaf, Carol D Jones, Birgitte Simen, Bozena Hanczaruk, Khoa D Nguyen, Kari

Nadeau, Michael Egholm, David B Miklos, James Zehnder, and Andrew Z Fire. Measurement and clinical monitoring of human lymphocyte clonality by massively parallel v-d-j pyrosequencing. *Science translational medicine*, 1:12ra23, 12 2009.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[10] Robert C. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[11] Mohammadreza Ghodsi, Bo Liu, and Mihai Pop. Dnaclust: accurate and efficient clustering of phylogenetic marker genes. In *BMC Bioinformatics*, 2011.

[12] Namita T. Gupta, Kristofor D Adams, Adrian W. Briggs, Sonia C. Timberlake, Francois Vigneault, and Steven H. Kleinstein. Hierarchical clustering can identify b cell clones with high confidence in ig repertoire sequencing data. *Journal of immunology*, 198 6:2489–2499, 2017.

[13] Ning Jiang et al. Lineage structure of the human antibody repertoire in response to influenza vaccination. *Science Translational Medicine*, 5(171), 2013.

[14] Chen Jin, Ruoqian Liu, Zhengzhang Chen, William Hendrix, Ankit Agrawal, and Alok Choudhary. A scalable hierarchical clustering algorithm using spark. In *Proceedings of the 2015 IEEE First International Conference on Big Data Computing Service and Applications*, BIGDATASERVICE '15, pages 418–426, 2015.

[15] Jacob Kogan. *Introduction to Clustering Large and High-Dimensional Data*. Cambridge University Press, 2006.

[16] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[17] Aaron Logan, Hong Gao, Chunlin Wang, Bita Sahaf, Carol D Jones, Eleanor L Marshall, Ismael Buo, Randall Armstrong, Andrew Z Fire, Kenneth I Weinberg, Michael Mindrinos, James Zehnder, Scott D Boyd, Wenzhong Xiao, Ronald Davis, and David B Miklos. High-throughput vdj sequencing for quantification of minimal residual disease in chronic lymphocytic leukemia and immune reconstitution assessment. *Proceedings of the National Academy of Sciences of the United States of America*, 108:21194–9, 12 2011.

[18] Keyue Ma, Chenfeng He, Ben S. Wendel, Chad M. Williams, Jun Xiao, Hui Yang, and Ning Jiang. Immune repertoire sequencing using molecular identifiers enables accurate clonality discovery and clone size quantification. *Frontiers in Immunology*, 9, 02 2018.

[19] Rognes T. Quince C. de Vargas C. Mahe, F. and M Dunthorn. Swarm: robust and fast clustering method for amplicon-based studies. *PeerJ*, 20014.

[20] Elaine Mardis. The impact of next-generation sequencing technology on genetics. *Trends in Genetics*, 24:133–141, 04 2008.

[21] Joo F. Matias Rodrigues and Christian von Mering. Hpc-clust: distributed hierarchical clustering for large sets of nucleotide sequences. *Bioinformatics*, 30(2):287–288, 2014.

[22] Antonio Mori, Sara Deola, Luciano Xumerle, Vladan Mijatovic, Giovanni Malerba, and Vladia Monsurr. Next generation sequencing: New tools in immunology and hematology. *Blood research*, 48:242–249, 12 2013.

[23] Clark F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

[24] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[25] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[26] Raghu Ramakrishn Tian Zhang and Miron Livn. Birch: an efficient data clustering method for very large databases. In *SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data Pages 103-114*, 1996.

[27] Ben S Wendel, Chenfeng He, Mingjuan Qu, Di Wu, Stefany M Hernandez, Ke-Yue Ma, Eugene W. Liu, Jun Xiao, Peter D Crompton, Susan K Pierce, Pengyu Y. Ren, Keke Chen, and Ning Jiang. Accurate immune repertoire sequencing reveals malaria infection driven antibody lineage diversification in young children. In *Nature Communications*, 2017.

[28] Yu-Chang Wu, David Kipling, and Deborah Dunn-Walters. Age-related changes in human peripheral blood igh repertoire following vaccination. *Frontiers in immunology*, 3:193, 07 2012.

[29] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.

[30] Lei Yu and Yongjun Guan. Immunologic basis for long hcdr3s in broadly neutralizing antibodies against hiv-1. *Frontiers in immunology*, 2014.

[31] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd*

USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.