Scalable Sequence Clustering for Large-Scale Immune Repertoire Analysis

1st Prem Bhusal
Dept. of Computer Science and Engineering
Wright State University
Dayton, OH, USA
remp.bhusal@gmail.com

2nd AKM Mubashwir Alam
Department of Computer Science
Marquette University
Milwaukee, WI, USA
mubashwir.alam@marquette.edu

3rd Keke Chen
Department of Computer Science
Marquette University
Milwaukee, WI, USA
keke.chen@marquette.edu

4th Ning Jiang
Department of Bioengineering
University of Pennsylvania
Philadelphia, PA, USA
jnjiang@seas.upenn.edu

5th Jun Xiao
ImmuDX LLC
Austin, TX, USA
julianxiao@gmail.com

***Abstract*—The development of the next-generation sequencing technology has enabled systems immunology researchers to conduct detailed immune repertoire analysis at the molecular level that allows researchers to understand the healthiness of a patient's immune system. Recent studies have shown that the single-linkage clustering algorithm can give the best results for B cell clonality analysis – a critical type of immune repertoire sequencing (IR-Seq) analysis. Large sequence datasets (e.g., millions of sequences) are being collected to comprehensively understand how a specific person's immune system evolves over different stages of disease development. However, the classical single-linkage clustering algorithm does not scale well to such large sequence datasets. Surprisingly, no study has been done to address this scalability issue for immunology research and development. We study three different strategies to scale up the single-linkage algorithm for sequence data. They include (1) the approximate single-linkage algorithm enhanced with the non-Euclidean indexing methods, (2) the Spark-based single-linkage algorithm (SparkMST) that was originally designed for vector data and now modified for sequence data, and (3) a new tree-based sequence summarization approach – SCT that aims to reduce the data for single-linkage clustering with well-preserved clustering quality.**

We have implemented these approaches and experimented with real sequence datasets for B cell clonality analysis. (1) The index-enhanced hierarchical clustering algorithm (e.g., VPT-HC using the Vantage-Point tree for indexing) preserves the clustering quality very well while significantly reducing the time complexity. (2) The SCT approach serving as a preprocessing step can effectively reduce data size for clustering. The overall clustering, SCT followed by VPT-HC, is the fastest among the evaluated single-machine algorithms. However, this approach also slightly affects the clustering quality. (3) The SparkMST parallel algorithm scales out nicely and also gives exact single-linkage clustering results. However, SparkMST is tied to the single-linkage algorithm and cannot be extended to general hierarchical clustering algorithms. Although this study focused on the specific application area: the B cell clonality analysis, we believe other sequence data analysis problems may find the developed scalable techniques useful.

This research was partially funded by NIH 1R43AI136357-01A1.

***Index Terms*—clustering, sequence data, scalability, parallel processing, summarization, indexing**

I. Introduction

The human immune system works amazingly well in defending the body against attacks by "foreign" invaders. One of the essential 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 various antibodies that fight different antigens, e.g., bacteria or viruses. Without infection, the B cells already carry genes for producing 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 handle new kinds of antigens. It has been observed that in healthy human adults, about 7% of B cells are mutated [1], which means each activation may generate about 10^6 new types of Ig molecules. The mutation rate and the B cell clonality distribution may vary due to aging [2] and disease [3], [4]. 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 [5], [6] was available, which is often referred to as the Adaptive Immune Receptor Repertoire sequencing (AIRR-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 1 Giga base pairs [1]. A typical sequence for studying B cell mutation has about 100-300 base pairs, and one experiment may generate millions of sequences. Thus, AIRR-seq has become a popular tool for molecule-level immune systems research, including B cell clonality

[1] 1 Giga base pairs (Gb) = 10^9 base pairs; one base pair of DNA/RNA is represented by one of the four characters "ACGT".

analysis. In the VDJ recombination process (Figure 1), B cells randomly assemble different gene segments – known as variable (V), diversity (D), and joining (J) gene segments, to generate unique antibodies. The AIRR-seq method profiles this VDJ segment to understand the diversity and similarity between mutations. Researchers have been studying the clustering structures of VDJ segments in a person's B cells to identify the B cell clones, e.g., with the sequences extracted from the person's 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. [7] have shown that the single-linkage hierarchical clustering algorithm gives the best results among other algorithms for B cell clonality 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 [2]. The critical challenge of this clustering analysis is the number of profiled sequences can be very large, from hundreds of thousands to millions. However, the optimal single-linkage clustering methods still take $O(N^2)$ time complexity for N sequences (and other types of hierarchical clustering are even more expensive, $O(N^2 \log N)$).) – in practice, it may take several 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 in the foreseeable future that will generate much more sequences, any significant development of faster and scalable hierarchical clustering algorithms will benefit the whole systems immunology research community. It will also help other biomedical research domains that heavily depend on clustering analysis of large-scale sequence data.

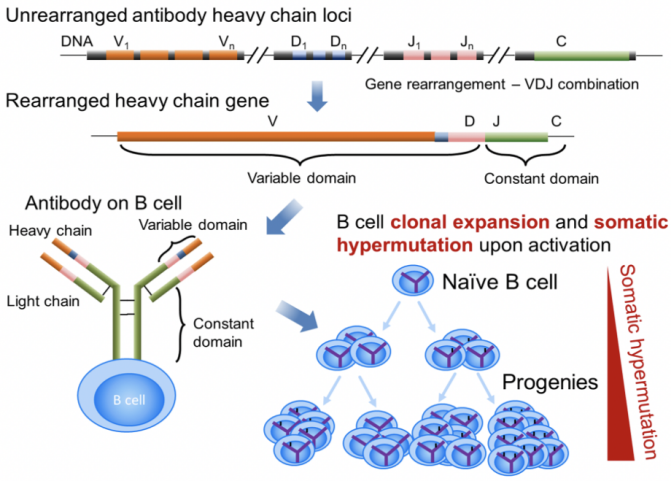

Fig. 1. Antibody repertoire generation and diversification during activation

We have developed an AIRR-Seq analysis pipeline (Section III), of which clustering analysis is an essential component. The framework takes sequencing results, cleans the data, assembles sequences, clusters them, and extracts the mutation information. The sizes of datasets for clustering analysis are around hundreds of thousands for current research problems [2] that involves only four time-points per patient. We expect larger datasets with more time points per patient to be generated, which may reach about a million sequences per person. To understand the costs of handling that scale of datasets, we have experimented with several strategies to speed up and improve the scalability of the single-linkage clustering methods for sequence datasets. This empirical study will adapt the existing approaches proven successful in other domains to the problem of clustering sequence data in AIRR-seq analysis. Specifically, we have developed the following strategies.

First, we will experiment with non-Euclidean indexing methods to reduce the complexity of key components in 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 reduce the time complexity of the initialization stage from $O(N^2)$ to $O(N \log N)$ and space complexity from $O(N^2)$ to $O(N)$. Many such indexing methods are only designed for the Euclidean space [8], often on low dimensional (2 or 3 dimensions) datasets for fast nearest neighbor search due to the curse of dimensionality [9]. It's unknown how effective the non-Euclidean indexing methods are for sequences, which use the similarity measures, such as edit distance, Hamming distance, and alignment-based distance [10], [11]. Our first strategy will investigate non-Euclidean indexing methods to speed up certain stages of hierarchical clustering and thus reduce the overall complexity.

Second, parallel processing can be effective for specific algorithm settings. There have been several efforts to develop the parallel or distributed version of the single-linkage algorithm. Olson [12] 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. [13] have implemented the minimum-spanning-tree (MST) algorithm with Apache Spark. Since single-linkage clustering is equivalent to the MST problem, the SparkMST works equivalently as a parallel single-linkage algorithm. As Gupta et al. [7] has shown that single-linkage clustering is the best for B cell clonality analysis, we consider that the SparkMST is a promising scalable multi-node solution for our application.

Third, we also explore the tree-based summarization approach as a preprocessing step for reducing the size of 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 [14] has shown such a tree works for numeric vector data on Euclidean distance space. However, no study has shown that this idea can also work for sequence data in non-Euclidean spaces and generate results comparable to single-linkage clustering for B cell clonality analysis.

We summarize our unique contributions in this important application of scalable hierarchical clustering techniques as follows.

- We have developed the Sequence Condensation Tree

(SCT) structure and algorithms for scanning the dataset once and absorbing sequences into a hierarchical summarization structure. It includes a novel design of node and tree structures and related algorithms, which can recursively summarize similar sequences under each tree branch. We show that this algorithm is highly scalable and resource-efficient (i.e., using only a single machine) if a slight quality loss is acceptable.
- We have experimented with a framework for integrating cluster indexing methods and cluster-level representative sequences with single-linkage clustering. It allows any non-Euclidean indexing method to work as a plugin, such as the VP-Tree [15], [16]. The result shows that this approach can preserve the clustering quality perfectly with good performance gain. If only a single machine is available, it can also be a good choice.
- We have modified and evaluated SparkMST [13] for processing sequence data in parallel. Our experimental result shows that SparkMST scales well to both the size of sequence data and the size of Spark cluster. It's a good candidate solution for users who can provide a large-scale Spark cluster. However, so far, the Spark-based algorithm is only available for single-linkage, not for other hierarchical clustering methods

The remaining sections of the paper are organized as follows. In Section 2, we give the basic notations and definitions. Section 3 briefly introduces our AIRR-Seq framework and shows that the sequence clustering analysis is the critical component of the framework. Section 4 describes the detailed strategies for fast and scalable sequence clustering, focusing on optimizing the single-linkage clustering method. Section 5 shows the experimental evaluation results. Finally, Section 6 covers some related work on sequence clustering, and Section 7 concludes our work.

II. Notations and Definitions

A sequence in our study is a string of characters formed from the nucleobases, namely {A, C, T, G}. The specific pairs of nucleobases can be bonded chemically, i.e., AT and CG, forming the two complementary strands in DNA. Sequencing techniques utilize this bonding mechanism [5], [6]. Therefore, the sequence length is often called the number of "base pairs". The typical sequence length for immune repertoire analysis is around 100-300, depending on the specific experimental design and study objectives. To unify the notations, we will use lower cases, e.g., m or n, to denote the sequence length and N to represent the number of sequences.

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

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. It's a popular choice for strings of unequal length. However, it's pretty expensive for long sequences. For two sequences of lengths m and n, respectively, the overall complexity of edit distance is $O(mn)$.

K-mer Based Distance: K-mer [10] refers to the substring of length k from the given string. Each sequence can be mapped to a set of core k-mers. The k-mer distance between any two sequences is defined as the number of unique k-mers shared by them. K-mer based distance can be used for approximate distance computation as it is computationally less expensive than other exact distance computations like edit distance and sequence alignment.

Sequence Alignment Based Distance: Sequence alignment is the process of arranging the DNA or protein sequences, which can also be used to define the sequence similarity. After an alignment, the sequence distance is defined as the number of mismatches between the two sequences. The popular alignment algorithms, such as Smith-Waterman [17] (with complexity $O(mn)$, are also expensive for long sequences.

Hamming Distance: Hamming distance is measured between two equal-length sequences with complexity $O(n)$ for the length n. It is defined as the number of positions at which corresponding characters differ. If we focus on small mutations typically seen in immune systems, i.e., base substitutions, Hamming distance can be a valid choice [7], which can significantly reduce the computational costs.

III. An AIRR-Seq Analysis Framework

We have developed a complete immune repertoire sequencing analysis framework and used it in several applications [2], [18]. Fig 2 gives the overall structure and different components of the 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 stages (1) and (2) are not very complex, and they scale well with the dataset. Reads processing has some well-known modules, such as read 1 and 2 alignments for paired-end sequencing, barcode removal, consensus building, and sequence truncating, which we skip the details [2]. Typically, the algorithms work on pairs of reads or individual sequences and are 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 parallel processing models like MapReduce [19]. After Lineage Formation, the lineage-level analyses [20] involves only thousands or tens of thousands of lineages, where scalability and speed are not an issue yet[2].

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 hundreds of thousands to millions). So far, the most effective algorithm for finding the B cell clones is single-linkage hierarchical clustering that generates the closest

[2]The scale may increase in the future for possibly different study objectives and depths.

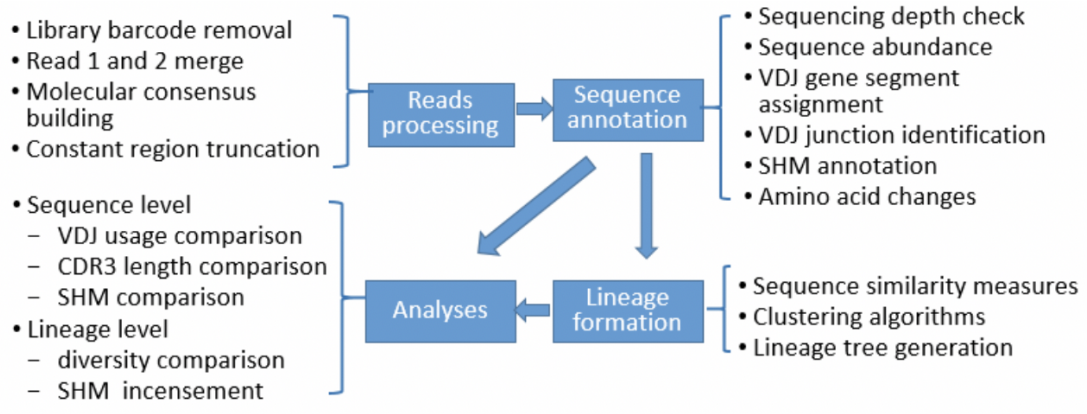

Fig. 2. Overview of our AIRR-Seq analysis framework

domain-specific clustering structures as shown by Gupta et al. [7]. Yet, its best complexity is $O(N^2)$ [12], which significantly restricts the scale of sequence datasets to be processed. Our goal is to investigate the methods that give (possibly approximate) single-linkage clustering structures with much faster processing time.

IV. Strategies for Scalable Single-Linkage Clustering Analysis of AIRR-Seq Data

Since single-linkage clustering gives the best results for B cell clonality analysis, our strategies for developing a scalable clustering algorithm will focus on this algorithm. However, we can also extend some strategies to general hierarchical clustering methods that biomedical researchers use in other domains. 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 supports fast nearest neighbor searches. The index-enhanced framework will be compared with a recently developed ESPRIT-tree [21] indexing for 16sRNA Sequence-based microbial community analysis. Second, we focus on the parallel implementation of minimum spanning trees, based on the SparkMST algorithm [13], as single-linkage clustering is equivalent to finding minimum spanning trees. Jin et al. have experimented with the Spark-based implementation for *vector data* up to two million records. However, there is no current study about its scalability on sequence data. We adopt the Spark MST algorithm and extend it to sequence data. Finally, we examine the idea of tree-based summarization as the preprocessing step and develop the SCT approach for processing the sequence data in one scan, which will be highly scalable even with only a single machine. We can then extract a smaller number of summary nodes for clustering. We will evaluate how close the generated clustering structure is to the ideal result.

In the following, we will give more details of the three strategies and conduct theoretical analyses to determine the potential performance gain.

A. Speeding up Sequence Clustering with Indexing and Cluster Representatives

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 Basic Sequence Agglomerative 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 can also index clusters of sequences. For the first requirement, we consider adopting existing indexing methods for general metric spaces, such as the Vantage-Point tree (VP tree) [15], [16] and Cover Tree [22]. For the second requirement, we use a representative-sequence method to approximately sketch the boundary of a cluster. Specifically, 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 grow after the merging operations. We use representative sequences in each cluster for approximate but faster general cluster-cluster distance computation (not only for single-linkage), which has preserved clustering quality satisfactorily. Traditionally, the cluster distance in hierarchical clustering is defined based on the pairwise distances between pairs of members from the two clusters, 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 working with the single-linkage merging steps. Assume we use k representatives to describe a cluster. (1) If the cluster contains one sequence, the 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 the set sizes be $|R_1| = n_1$ and $|R_2| = n_2$. If $n_1 + n_2 > k$, 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 "virtual" location of the sequence inside the cluster: the ones with the largest sums are around the boundary of the cluster. We will maintain the top k candidates with 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 among all candidates. The overall update cost is $O(k^2)$.

The distance between a cluster C and a sequence s can be approximately computed, based on the distances between representatives and the sequence. Specifically, for the single-linkage algorithm, it's $\min_{r_i \in RofC} distance(r_i, s)$. Similarly, the distance between clusters is translated to $\min_{r_i \in R_1, r_j \in R_2} distance(r_i, r_j)$ for all reprentatives r_i from C_1 and r_j from C_2. After merging a pair of clusters C_1 and its nearest neighbor C_2, C_1 and C_2 are moved from the index and the merged cluster $C_{1,2}$ with its updated representatives are inserted back to the index using the cluster-sequence and cluster-cluster distance computation methods.

Cost Analysis. Ideally, the initial index building time is about $O(N \log N)$, and the initial setup of nearest neighbors costs $O(N \log N)$ distance computations. The steps in the clustering loop involve $(\log N)$ nearest neighbor search and $O(k^2 \log N)$ distance computations to update the index. Thus, with the help of indexing, we can reduce the number of overall expensive distance computations to $O(k^2 N \log N)$. We will evaluate the costs in Section V with the VP-tree as the indexing structure and compare with another index-enhanced sequence clustering method ESPRIT-tree [21] developed for 16s rRNA sequence analysis.

B. Parallel Single-Linkage Clustering

The second candidate solution is to scale up the processing with a parallel processing computing cluster, e.g., the Spark [23] based approach. The hope is to develop an algorithm that speeds up reasonably with the increased size of the Spark cluster. There have been several efforts to design parallel or distributed versions of the single-linkage hierarchical clustering algorithm in the past [12]. 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. [13] have developed a parallel minimum spanning tree (MST) with Spark, which can be used to generate equivalent single-linkage hierarchical clustering results.

The MST method can be parallelized in two stages in SparkMST. 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 between the complete graphs. To avoid computing all edge weights, the first stage of SpartMST uses Prim's algorithm to find the MSTs of the subgraphs. In the second stage, Kruskal's algorithm works efficiently with only the extracted edges from the MSTs generated in the first stage.

Cost and Scalability Analysis. The complexity of the basic Prim's algorithm is $O(N^2 \log N)$ for dense graphs ($E = O(N^2)$) with the help of priority queue for N sequences [3]. 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 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 out with the number of parallel workers, w. The output of Prim's algorithm will contain $O(N/p)$ edges for both the complete graph and the bipartite graph cases, respectively. Thus, there are $O(p^2)$ of such edge sets. With Kruskal's algorithm, we can progressively merge pairs of edge sets 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 in our study. The experimental evaluation also shows its good scalability. While the performance gain might be minor with a small number of computing nodes, a larger Spark cluster does help reduce the cost. The only limitation is that it only works as single-linkage

[3] With Fibonacci heaps, the cost can be further optimized to $O(N^2 + \log N)$

clustering. To our knowledge, no parallel solutions for general hierarchical clustering yet.

C. Sequence Condensation Tree (SCT) – a fast sequence summarization algorithm

We develop the SCT fast sequence summarization algorithm that scans the dataset only once and sketches the small clusters in the dataset, based on the idea of BIRCH [14] that works only for vector data, however. 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 computations. Thus, the cost of processing one sequence is fast, at the level of $O(\log N)$.

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. Creating a new entry may cause the node to split when the predefined maximum number of entries for the node is reached. Node split may be propagated level-by-level up to the root to balance the tree, a similar process used by multi-way balanced trees. The core similarity computation is defined between node and sequence. If we treat a node as a cluster, we can use the representative-sequence method defined previously in Section IV-A. While a new sequence is inserted into the leaf node, we use the total sum method to check whether this sequence can be a new representative sequence for that node. Based on the existing 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 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 sequences and the inserted sequence to define the node-sequence distance. Min performs similar to the single-linkage merging criterion.

A simple structure of SCT is illustrated in Figure 3. The SCT has three 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 clonality analysis, we don't need to separate very similar sequences, e.g., those with their similarity higher than 95% of the sequence length merged into one leaf entry.

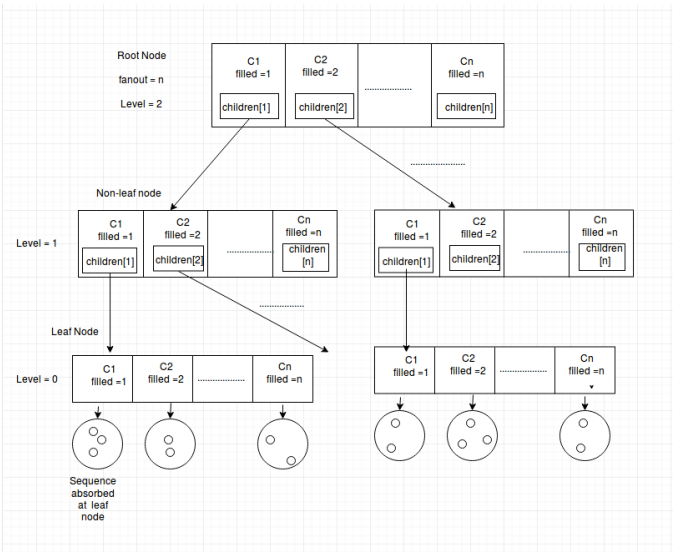

Fig. 3. Structure of Sequence Condensation Tree (SCT)

As a preprocessing and data reduction method, SCT has several unique advantages for clustering sequence data.

- 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 tree's height, which is $O(\log_f N)$ for N sequences, where f is the number of entries per node. For large f, e.g., f=10 or 20, the height grows slowly.
- 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 single-linkage hierarchical structures based on the selected nodes.

V. Experiments

The goals of the experimental evaluation include: (1) the scalability of single-machine processing methods: the index-enhanced single-linkage approach, and the impact of SCT data reduction, (2) the clustering quality of the single-machine approximate methods, and (3) the scalability of multi-node Spark-based method: SparkMST.

A. Setup

We have implemented all the discussed algorithms for sequence data. VP-tree is used as the indexing method for the index-enhanced single-linkage approach, denoted as VPT-HC. The VPT-HC can be applied to the SCT reduced datasets, and we name this approach SCT-VPT-HC. We compare this approach with a somewhat close approach ESPRIT-Tree [21] developed for clustering microbial rRNA sequences in analyzing the microbial communities. The ESPRIT-Tree implementation is available as a compiled binary from C++ source code. We implemented all our algorithms with Scala.

The single-machine algorithms are evaluated on a Linux server equipped with Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz and 32GB memory. For parallel processing, we have used a standalone cluster with 11 powerful nodes (1 head node + 10 slave nodes), each of which has 16 cores and 250 GB memory. Thus, we can set up a Spark cluster with up to 160 parallel workers.

B. Quality Measures and Baseline Establishment

Since the recent study [7] has shown that the single-linkage algorithm is the best option for B cell clonality 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) cannot handle the scale of our sequence data, e.g., requiring the whole distance matrix as the input. We have used our implemented SparkMST algorithm to generate the baseline.

The ground truth clusters are defined based on the minimum spanning tree generated with SparkMST. Systems immunology researchers [7], [18] have adopted the distance-to-nearest-neighbor method for finding the heuristic distance threshold for cutting the hierarchical clustering result. Specifically, in B cell clonality analysis, the histogram of sequences' distances to their nearest neighbors has shown an approximate bimodal pattern in Figure 4, where the x-axis is the normalized distances, i.e., distances/max_sequence_length, in the range $[0, 1]$ and the y-axis is the number of sequences having the certain distance to their nearest neighbors. The first mode is mapped to the sequences within their clusters, and the second mode implies the outliers (or singletons). The ideal distance cut is located at the valley between the first two modes. Previous studies show the values are around 0.07-0.15 for normalized distances, although different datasets have slightly different values. For simplicity, we have used 0.12 in experiments to establish the baseline clusters. Note that this heuristic is used for B cell clonality analysis only, which may not fit other applications of hierarchical clustering.

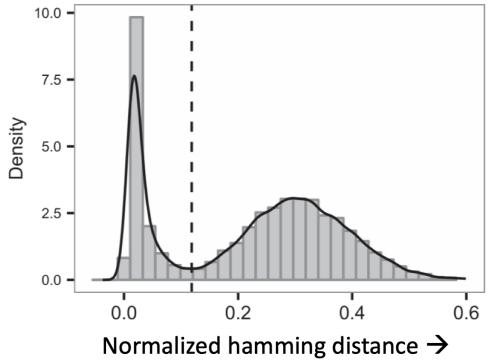

Fig. 4. Threshold determination by distance-to- nearest-neighbor method (Credit [7]).

With the baseline labeled clusters set up, we use Normalized Mutual Information(NMI) for determining the quality of a given clustering result. Let the ground truth cluster labels be the "true labels" drawn from a random variable T, and the clustering algorithm assignments be the "predicted labels" randomly 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

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

C. Datasets

We have evaluated the algorithms with two datasets:

- *Dataset 1*: This dataset is shared by Gupta et al. [7]. It contains 1.1 million sequences of variable length. The dataset is available on the Sequence Read Archive [4] under Bio- Project accession numbers SRX2018085 under PRJNA338795 (study of B cell repertoire in myasthenia gravis). The average length is about 100 base pairs. Large groups of equal-length sequences are used in experiments.
- *Dataset 2*: This dataset is used in studying the B cell clonality of Malaria patients [2]. 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. The dataset consists of around 164 thousand sequences with an equal length of 270 base pairs.

D. Result and discussion

Clustering Quality. We first show the clustering quality of the single-machine approximation methods based on the NMI measure, using the SparkMST result as the reference. As SparkMST generates the exact result of single-linkage clustering, we don't need to evaluate the clustering quality and instead focus on its scalability in later discussion. The VP-Tree enhanced single-linkage algorithm (VPT-HC) generates the clustering dendrogram, and then we use the bimodal distance threshold to cut the dendrogram and get the clusters. SCT is used to generate the summary tree with leaf thresholding of ≤ 0.05 normalized distance for absorbing highly similar sequences and the fanout 10 to achieve a relatively flat tree. For simplicity, we collect only the SCT leaf nodes for single-linkage clustering, which we use VPT-HC. The quality results for all the algorithms are present on Table I for Dataset 1 and on Table II for Dataset 2. "SCT-VPT-HC center/avg/min" represent different node-sequence distance computation methods used by the SCT method. Each table has two columns corresponding to different sequence distance methods for SCT-VPT-HC and VPT-HC, while ESPRIT-Tree uses a k-mer based similarity measure [21].

Both Table I and Table II show that VP-Tree enhanced algorithm (VPT-HC) has the highest NMI score. For Dataset2, the quality of VPT-HC is almost perfect. The ESPRIT-Tree and

[4]https://www.ncbi.nlm.nih.gov/sra

TABLE I
NMI SCORES ON DATASET 1. 0 <= NMI <= 1. THE LARGER, THE BETTER.

Method	Edit Dist.	Hamming Dist.
VPT-HC	0.96 ±0.04	0.97± 0.04
SCT-VPT-HC(center)	0.93±0.03	0.94± 0.03
SCT-VPT-HC(avg)	0.93 ±0.04	0.94±0.04
SCT-VPT-HC(min)	0.92 ± 0.03	0.94± 0.02
ESPRIT-Tree	0.93 ± 0.04	(with k-mer similarity [21])

TABLE II
NMI SCORES FOR DATASET 2

Method	Edit Dist.	Hamming Dist.
VPT-HC	0.995 ± 0.003	0.996± 0,002
SCT-VPT-HC(center)	0.92 ± 0.01	0.95± 0.01
SCT-VPT-HC(avg)	0.93 ±0.09	0.95± 0.01
SCT-VPT-HC(min)	0.90 ± 0.01	0.93± 0.01
ESPRIT-Tree	0.95± 0.00	

SCT-VPT-HC methods of different settings have similar performance, slightly lower than VPT-HC's. Another interesting observation from this result is that Hamming distance works equally well as Edit distance, which is consistent with the conclusion of the previous study [7]. Thus, in the following, we will use Hamming distance for scalability experiments.

Scalability of single-machine algorithms. To evaluate the computational costs of the single-machine algorithms and their scalability, we sampled both datasets to get different sizes ranging from 10 thousand to 60 thousand. Hamming distance is used for clustering. In Figure 5 and 6, the x-axis represents the sequence size in thousands, and the y-axis represents the time cost in seconds. We used SCT-VPT-HC (center) for this evaluation. For the same size of data, Dataset2 is more expensive due to its longer sequences. The result shows that VPT-HC and ESPRIT-Tree have a similar growth pattern, slightly higher than linear complexity. In comparison, SCT-VPT-HC is much faster because SCT helps significantly reduce the data size by about 62% for Dataset 1 and 51% for Dataset 2. This size reduction is achieved at a low cost. Figure 7 shows that the SCT summarization part occupies a relatively small portion of the overall cost of SCT-VPT-HC. With carefully tuned cutting strategies to get an even smaller number of non-leaf nodes, we think the overall SCT-VPT-HC cost can be further reduced. However, as we have known earlier, the fast speed also comes at the expense of slightly reduced clustering quality.

Scalability of SparkMST on sequence data. We study the scalability of SparkMST in two settings: how the algorithm scales with the increasing number of sequences for a fixed number of computing nodes and how it scales with the increasing number of computing nodes for the same data size. We use a ten-node cluster setup for the first experiment. Then, we vary the Spark cluster configuration with 2, 4, 6, 8, and 10 nodes in the second experiment. Again, Hamming distance is used in clustering.

Figure 8 shows the costs for processing different sizes of data by comparing SparkMST and SCT-VPT-HC. The previously evaluated SCT-VPT-HC is used for reference. With a 10-node cluster, SparkMST has significantly lower costs and scales much better than SCT-VPT-HC. Next, we study how the algorithm scales with the increasing number of computing nodes while the data size remains unchanged. It allows us to understand whether the algorithm can *scale out* nicely by adding more computational nodes. We get 160 thousand sequences from each dataset for this study. Figure 9 shows that the algorithm scales out pretty well. The classical "speedup" measure is defined as "the time cost with one node / the time cost with n nodes". Ideally, for n nodes, the speedup is n. By looking at Figure 10, we can also clearly see the speedup increasing steadily with the increased nodes. From two to six nodes, the speedups are almost ideal. In general, for a larger number of nodes, one can expect that higher extra costs are spent on scheduling and communications, which normally reduce the speedup. The speedup stays at about eight for ten nodes, which is still considered excellent scalability in parallel processing.

Discussion. Based on the evaluation results, we can conclude that all the three approaches work as expected with different levels of performance gain. First, the index-enhanced methods can effectively reduce the cost of the original single-linkage clustering for the single-machine setting. However, we observed that the current implementations are still quite expensive – processing 60k sequences takes about three hours. We will experiment with other non-Euclidean indexing structures, such as Cover Tree [22], to improve the performance of this method. In comparison, the SCT summarization method can effectively reduce the data size, thus significantly reducing the cost of clustering. However, the clustering quality is also slightly lower than the indexing-enhanced methods on the whole dataset. We will investigate the SCT's post-processing algorithms to reduce the cost further while maintaining good clustering quality. Second, SparkMST gives exact single-linkage clustering results. It is still expensive for small-scale Spark clusters. For example, Figure 9 shows its costs on the two-node cluster are still around 4000 seconds for Dataset1 and 8000 seconds for Dataset2. However, the algorithm scales well for an increasing number of computing nodes. SparkMST on a ten-node cluster runs much faster than SCT-VPT-HC and scales better for large data sizes. Therefore, our recommendation is to use SparkMST if the user has an abundant budget for a large Spark cluster. If only a single server is available, one can use either the SCT method for fast preprocessing but slightly lower clustering quality or index-enhanced single-linkage with more time but a better quality guarantee. However, we think the SCT summarization method may have broader applications for other more sophisticated clustering algorithms such as spectral clustering. Finally, the Spark-based solution is only available for single-linkage, not other hierarchical clustering algorithms. In contrast, the indexing-enhanced method can be easily extended for different types of hierarchical clustering.

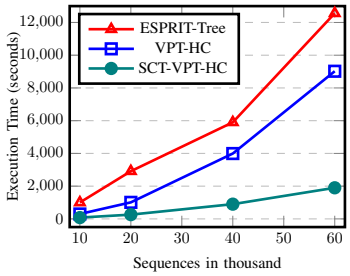

Fig. 5. Cost comparison between different methods on data sampled from Dataset1 for single-machine algorithms.

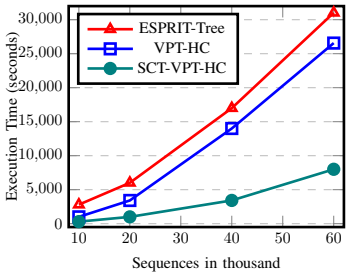

Fig. 6. Cost comparison between different methods on data sampled from Dataset2 for single-machine algorithms.

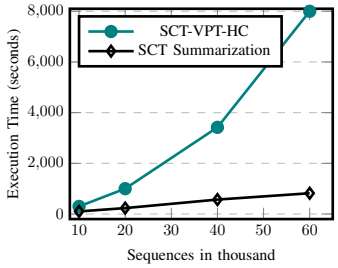

Fig. 7. The cost of SCT summarization occupies a small portion in the overall SCT-VPT-HC cost (Dataset2).

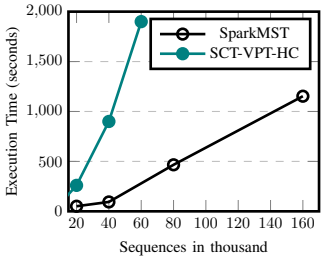

Fig. 8. Scalability of SparkMST on a ten-node Spark cluster with increased data size on Dataset 1.

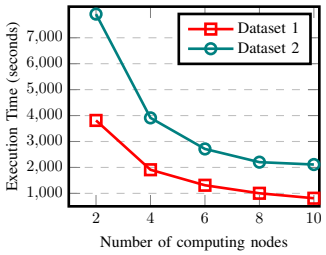

Fig. 9. Scalability of SparkMST with the increased number of computing nodes and a fixed data size

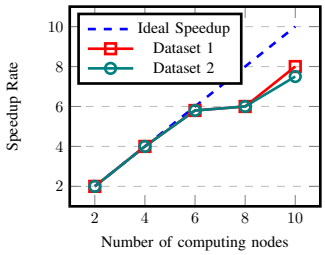

Fig. 10. Scalability of SparkMST: speedups according to the number of nodes

VI. Related Work

Many well-known fast sequence clustering algorithms, such as CD-HIT [10], UCLUST [24], DNACLUST [11], are based on simple threshold-based grouping, which does not give the ideal clustering structures. CD-HIT [10] 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 – 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 $O(Nn)$ complexity where $n < N$, n is the number of 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 [24] is similar to CD-HIT [10] and thus has similar complexity. It uses a fast sequence search heuristic to find the closest seed sequence, which helps when n grows large. DNACLUST [11], another similar threshold-based approach, uses edit distance as the similarity measure. This category of methods only works 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 clustering result is also highly dependent 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 still have $O(N^2 \log N)$ complexity, which does not scale well for large sequence datasets. Biomedical researchers have been using the expensive algorithms, enduring long running times, until data size grows to an unacceptable level. Recently, Cai et al. proposed the ESPRIT-Tree [21] method, aiming to address the performance issues with million-level sequence sets. It uses k-mer based distance computation and a leveled tree to organize sequences. Each layer in the tree has a specific lay-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 the 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 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 [25] tried a multi-threading approach to achieve parallelization in the distance calculation step, which does not address the scalability bottleneck in the clustering steps. For the specific type of hierarchical algorithms: the single-linkage algorithm, Olson [12] has shown that the $O(N)$ 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 particular architectures are not accessible to most biomedical researchers. Recently, Jin et al. [13] proposed 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 the scaling-out performance is not so satisfactory.

The tree-based summarization methods have been applied in data streaming processing, e.g., the BIRCH method for vector data in Euclidean space [14], [26] and the HE-Tree method for categorical data [27]. However, to our knowledge, no work is reported on sequence data. We design the SCT approach for sequence data and show that it has low processing costs and is highly scalable for large datasets. Downstream analytics tasks, such as clustering, can work with a much smaller set of summary nodes than the original dataset.

VII. Conclusion

Recent studies have shown that single-linkage clustering is the best method for B cell clonality analysis. However, the scale of sequence data for B cell clonality analysis is rapidly increasing to a level that the classical single-linkage algorithm cannot comfortably handle anymore. We have studied three strategies to scale up/out the single-linkage clustering algorithm for large sequence datasets: the index-enhanced single linkage, the SparkMST parallel algorithm, and the SCT fast sequence summarization algorithm for reducing data in preprocessing. We have done an extensive experimental evaluation on two real B cell repertoire sequence datasets. The result shows that the index-enhanced single linkage can preserve the clustering quality almost perfectly. The SCT-assisted approach can significantly speed up the clustering process if only a single server is available. The SparkMST can scale well if one can afford a large computing cluster, but does not apply to other hierarchical clustering algorithms that users in other domains may want to try. Our study is an excellent practical example showing how big data techniques can help address real scalability problems in scientific applications and make it possible for scientists to discover new knowledge faster from larger datasets.

VIII. Acknowledgments

This research was partially supported by the National Institute of Allergy and Infectious Diseases under award number 1R43AI136357-01A1. The content is solely the authors' responsibility and does not necessarily represent the official views of the National Institutes of Health.

References

[1] Y.-C. Wu, D. Kipling, and D. Dunn-Walters, "Age-related changes in human peripheral blood igh repertoire following vaccination," *Frontiers in immunology*, vol. 3, p. 193, 07 2012.

[2] B. S. Wendel, C. He, M. Qu, D. Wu, S. M. Hernandez, K.-Y. Ma, E. W. Liu, J. Xiao, P. D. Crompton, S. K. Pierce, P. Y. Ren, K. Chen, and N. Jiang, "Accurate immune repertoire sequencing reveals malaria infection driven antibody lineage diversification in young children," in *Nature Communications*, 2017.

[3] S. D Boyd *et al.*, "Measurement and clinical monitoring of human lymphocyte clonality by massively parallel v-d-j pyrosequencing," *Science translational medicine*, vol. 1, 12 2009.

[4] A. Logan, H. Gao, C. Wang, B. Sahaf, C. D Jones, E. L Marshall, I. Buño, R. Armstrong, A. Z Fire, K. I Weinberg, M. Mindrinos, J. Zehnder, S. D Boyd, W. Xiao, R. Davis, and D. 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*, vol. 108, 12 2011.

[5] A. Mori, S. Deola, L. Xumerle, V. Mijatovic, G. Malerba, and V. Monsurrò, "Next generation sequencing: New tools in immunology and hematology," *Blood research*, vol. 48, pp. 242–249, 12 2013.

[6] E. Mardis, "The impact of next-generation sequencing technology on genetics," *Trends in Genetics*, vol. 24, pp. 133–141, 04 2008.

[7] N. T. Gupta, K. D. Adams, A. W. Briggs, S. C. Timberlake, F. Vigneault, and S. H. Kleinstein, "Hierarchical clustering can identify b cell clones with high confidence in ig repertoire sequencing data," *Journal of immunology*, vol. 198, no. 6, pp. 2489–2499, 2017.

[8] R. Sibson, "Slink: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

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

[10] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[11] M. Ghodsi, B. Liu, and M. Pop, "Dnaclust: accurate and efficient clustering of phylogenetic marker genes," in *BMC Bioinformatics*, 2011.

[12] C. F. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Computing*, vol. 21, no. 8, pp. 1313–1325, 1995.

[13] C. Jin, R. Liu, Z. Chen, W. Hendrix, A. Agrawal, and A. Choudhary, "A scalable hierarchical clustering algorithm using spark," in *Proceedings of the IEEE Conference on Big Data Computing Service and Applications*, 2015, pp. 418–426.

[14] R. R. Tian Zhang and M. Livn, "BIRCH: an efficient data clustering method for very large databases," in *ACM SIGMOD Conference on Management of data Pages 103-114*, 1996.

[15] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1993.

[16] T. Bozkaya and Z. Ozsoyoglu, "Indexing large metric spaces for similarity search queries," *ACM Trans. Database Syst.*, vol. 24, pp. 361–404, 09 1999.

[17] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[18] N. Jiang *et al.*, "Lineage structure of the human antibody repertoire in response to influenza vaccination," *Science Translational Medicine*, vol. 5, no. 171, 2013.

[19] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*, 2004, pp. 137–150.

[20] K. Chen, V. S. A. Gogu, D. Wu, and J. Ning, "Colt: Constrained lineage tree generation from sequence data," in *IEEE Conference on Bioinformatics and Biomedicine (BIBM)*, 2016, pp. 102–106.

[21] Y. Cai and Y. Sun, "Esprit-tree: hierarchical clustering analysis of millions of 16s rrna pyrosequences in quasilinear computational time," *Nucleic Acids Research*, vol. 39, no. 14, 2011.

[22] A. Beygelzimer, S. Kakade, and J. Langford, "Cover tree for nearest neighbor calculations," 2006.

[23] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the USENIX Conference on Hot Topics in Cloud Computing*. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10.

[24] R. C. Edgar, "Search and clustering orders of magnitude faster than blast," *Bioinformatics*, vol. 26, no. 19, pp. 2460–2461, 2010.

[25] J. F. Matias Rodrigues and C. von Mering, "Hpc-clust: distributed hierarchical clustering for large sets of nucleotide sequences," *Bioinformatics*, vol. 30, no. 2, pp. 287–288, 2014.

[26] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *International Conference on Very Large Databases*, 2003, pp. 81–92.

[27] K. Chen and L. Liu, "HE-Tree: a framework for detecting changes in clustering structure for categorical data streams," *VLDB Journal*, vol. 18, 2009.