Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Original Paper

Sequence analysis

A parallel computational framework for ultra-large-scale sequence clustering analysis

Wei Zheng[1], Qi Mao[2], Robert J. Genco[3], Jean Wactawski-Wende[4], Michael Buck[5], Yunpeng Cai[6] and Yijun Sun[1,2,*]

[1]Department of Computer Science and Engineering, [2]Department of Microbiology and Immunology, [3]Department of Oral Biology, [4]Department of Epidemiology and Environmental Health, [5]Department of Biochemistry, University at Buffalo, The State University of New York, Buffalo, USA and [6]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on October 19, 2017; revised on June 14, 2018; accepted on July 6, 2018

Abstract

Motivation: The rapid development of sequencing technology has led to an explosive accumulation of genomic data. Clustering is often the first step to be performed in sequence analysis. However, existing methods scale poorly with respect to the unprecedented growth of input data size. As high-performance computing systems are becoming widely accessible, it is highly desired that a clustering method can easily scale to handle large-scale sequence datasets by leveraging the power of parallel computing.
Results: In this paper, we introduce SLAD (**S**eparation via **L**andmark-based **A**ctive **D**ivisive clustering), a generic computational framework that can be used to parallelize various *de novo* operational taxonomic unit (OTU) picking methods and comes with theoretical guarantees on both accuracy and efficiency. The proposed framework was implemented on Apache Spark, which allows for easy and efficient utilization of parallel computing resources. Experiments performed on various datasets demonstrated that SLAD can significantly speed up a number of popular *de novo* OTU picking methods and meanwhile maintains the same level of accuracy. In particular, the experiment on the Earth Microbiome Project dataset (∼2.2B reads, 437GB) demonstrated the excellent scalability of the proposed method.
Availability and implementation: Open-source software for the proposed method is freely available at `https://www.acsu.buffalo.edu/~yijunsun/lab/SLAD.html`.
Contact: yijunsun@buffalo.edu
Supplementary information: Supplementary data is available at *Bioinformatics* online.

1 Introduction

Microbes play an essential role in processes as diverse as human health and biogeochemical activities critical to life in all environments on earth. However, due to the inability of traditional techniques to cultivate most microbes, our understanding of complex microbial communities is still very limited. The advent of high-throughput sequencing technology allows researchers to study genetic materials recovered directly from natural environments and opens a new window to extensively probe the hidden microbial world. Consequently, metagenomics, where the amplicon sequencing of 16S rRNA gene serves as a major probing tool, has recently become an exploding research area and was selected as one of the ten technical breakthroughs in 2013 by the Science magazine (Editorial, 2013).

In 16S rRNA sequence analysis, the first major step after quality control is usually to bin sequences into taxonomic or genotypic units, which forms the basis for performing ecological statistics and comparative studies (Di Bella *et al.*, 2013; Sun *et al.*, 2010). Existing methods can be generally classified into taxonomy-dependent approaches, where sequences are annotated against a reference database, and taxonomy-independent approaches (Mande *et al.*, 2012), where sequences are clustered into operational taxonomic units (OTUs) based on pairwise similarities without using external references (thus also called *de novo* OTU picking). Since the main goal of metagenomic studies is to explore uncharted biospheres where a significant portion of genetic material is contributed by previously unknown taxa, taxonomy-independent analysis is often the preferred, if not the only, choice.

A dozen of methods have been proposed in the last decade for *de novo* OTU picking of 16S rRNA sequences (Li and Godzik, 2006; Edgar, 2010; Sun *et al*., 2009; Cai and Sun, 2011; Schloss and Handelsman, 2005; Ye, 2010; Cai *et al*., 2017). Yet, the computational burden of generating clusters from massive sequence data remains a serious challenge, and only a few algorithms are able to handle millions of sequences. Based on the structure into which generated OTUs are organized, existing methods can be generally divided into two categories: hierarchical clustering (HC) and greedy heuristic flat clustering. HC is one of the most widely used approaches for sequence binning (Sun *et al*., 2009; Cai and Sun, 2011; Di Bella *et al*., 2013). It organizes sequences in a hierarchical tree, enabling researchers to examine OTUs at various similarity levels that may bear biological significance. A major drawback of HC is its extremely high computational complexity stemming mainly from the need of computing and storing a pairwise distance matrix, making it unsuitable for large-scale sequence analysis. Various data pre-processing heuristics have recently been proposed that were proven to be very effective in reducing the computational complexity of a clustering process (Schloss and Westcott, 2011). Yet, these heuristics do not fundamentally change the nature that HC is an $\mathcal{O}(N^2)$ algorithm. As a trade-off between computational efficiency and accuracy, several heuristic methods including Cd-hit (Li and Godzik, 2006) and UCLUST (Edgar, 2010) were proposed that employ greedy flat clustering to reduce the computational complexity associated with sequence binning. The basic idea is to process input sequences sequentially, by either assigning each sequence to an existing cluster or designating it as the center of a new cluster if the distances between the sequence and the centers of all existing clusters are larger than a pre-defined threshold. As such, heuristic methods calculate only the distances between input sequences and cluster centers, and run much faster than hierarchical clustering, though at the cost of decline of clustering quality (Chen *et al*., 2013; Schloss and Westcott, 2011; Sun *et al*., 2011). However, the sizes of OTUs generally exhibit a long-tailed distribution (Sun *et al*., 2011), meaning that there are a few large OTUs and a large number of small OTUs, and when processing massive sequence data, the number of OTUs is non-negligible. Consequently, existing heuristic methods are still not sufficient to handle extremely large datasets.

As high-performance computing systems are becoming widely accessible, it is highly desired that a clustering method can easily scale to handle massive sequence data by leveraging the power of parallel computing. However, efficient parallelization of sequence clustering is inherently difficult. For example, for UCLUST and Cd-hit, distance calculation in each iteration depends on cluster centers generated in previous iterations; for hierarchical clustering, each merging or dividing operation relies on the results of all previous merging or dividing operations. Several attempts, including HPC-CLUST (Matias Rodrigues and von Mering, 2013), DACE (Jiang *et al*., 2016) and subsampled open-reference clustering (Rideout *et al*., 2014), have been made to speed up a clustering process by utilizing the power of parallel computing. However, existing methods do not sufficiently address the computational issue associated with large-scale sequence analysis. HPC-CLUST takes a pre-aligned profile as input, which is computationally very expensive to calculate. For DACE, data partition relies on locality sensitive hashing (LSH) for approximate nearest neighbor search. While there exist a number of hash functions designed for various similarity measures (Slaney and Casey, 2008), it remains an open problem to perform LSH on sequence alignment distances. Subsampled open-reference clustering first generates cluster centroids by using randomly sampled sequences and then assigns remaining sequences to the centroids in parallel. However, it did not fundamentally address the issue of clustering parallelization, since the clustering process performed on sampled sequences remains a single-thread procedure and becomes the performance bottleneck when the number of sequences becomes excessively large.

```
Algorithm 1: SLAD($\mathcal{D}$, $\mathcal{A}$, $\mathcal{B}$, $k$)
  Input: data $\mathcal{D}$, clustering method $\mathcal{A}$, OTU picking method $\mathcal{B}$, the
         number of division branches $k$
  Output: hierarchical tree $\mathcal{T}_{slad}$
  run $\mathcal{T}_{slad}$ = LADC($\mathcal{D}$, $\mathcal{A}$, $k$);
  extract leaf nodes of $\mathcal{T}_{slad}$ as $\{\mathcal{C}_1, \cdots, \mathcal{C}_t\}$;
  for $i = 1$ to $t$ do
      $\mathcal{T}_i$ = $\mathcal{B}(\mathcal{C}_i)$;
      replace the $i$th leaf node of $\mathcal{T}_{slad}$ with the root node of $\mathcal{T}_i$;
  end
```

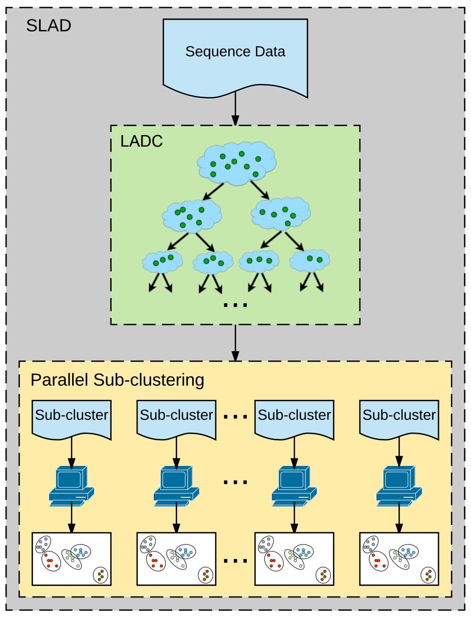


Fig. 1. Overview of the proposed parallel computational framework for ultra-large-scale sequence clustering analysis. LADC: landmark-based active divisive clustering.

In this paper, we proposed a general-purpose computational framework referred to as SLAD (**S**eparation via **L**andmark-based **A**ctive **D**ivisive clustering) that can in principle be used to parallelize any single-thread *de novo* OTU picking method. Theoretical analysis was performed that showed that the proposed method has a linearithmic computational complexity and can recover the true clustering structure with a high probability under some mild assumptions. We implemented the proposed method on Apache Spark, which allows us to easily and fully utilize parallel computing resources. Experiments performed on various datasets demonstrated that SLAD can significantly speed up a number of commonly used *de novo* OTU picking methods while maintaining the same level of accuracy. Finally, we conducted a scalability study on the Earth Microbiome Project (EMP) dataset (Gilbert *et al*., 2014) ($\sim$2.2B reads, 437GB). To our knowledge, this is the largest *de novo* OTU picking analysis ever performed in a distributed computing environment. By using 17 computer processors provided by Amazon Cloud, our method coupled with UCLUST finished the analysis of the 2.2B sequences in $\sim$17.8 hours. In contrast, it was estimated that it would take UCLSUT $\sim$636 days to finish the analysis on a single computer.

Algorithm 2: LADC($\mathcal{D}$, $\mathcal{A}$, k)

```
Input: data $\mathcal{D}$, clustering method $\mathcal{A}$, the number of division
    branches $k$
Output: hierarchical divisive tree $\mathcal{T}_{ladc}$
if termination conditions are satisfied then
    return $\mathcal{D}$;
end
$s = \log_2|\mathcal{D}|$;
$q = |\mathcal{D}|/k$;
choose $l_0 \in [1, |\mathcal{D}|]$ uniformly at random;
$\mathcal{L} = \{l_0\}$;
$d_{min}[\ i\ ] = d(x_{l_0}, x_i),\ \forall i \in [1, |\mathcal{D}|]$;
for $iter = 1$ to $s - 1$ do
    obtain indexes $i_1, \cdots, i_{|\mathcal{D}|}$ by sorting $d_{min}$ in descending
    order;
    choose $l_{iter} \in \{i_1, \cdots, i_q\}$ uniformly at random;
    $\mathcal{L} = \mathcal{L} \cup \{l_{iter}\}$;
    $d_{min}[\ i\ ] = \min(d_{min}[\ i\ ], d(x_{l_{iter}}, x_i)),\ \forall i \in [1, |\mathcal{D}|]$;
end
$\mathcal{S} = \{x_l \in \mathcal{D} \mid \forall l \in \mathcal{L}\}$;
obtain $\{\tilde{\mathcal{C}}_i\}_{i=1}^k$ by applying $\mathcal{A}(\mathcal{S}, k)$;
for $x_j \in \mathcal{D} \setminus \mathcal{S}$ do
    find the closest cluster $\tilde{\mathcal{C}}_m$ of $x_j$
    $m = \arg\min_{i=1,\cdots,k} \frac{1}{|\tilde{\mathcal{C}}_i|} \sum_{x_l \in \tilde{\mathcal{C}}_i} d(x_j, x_l)$;
    $\mathcal{C}_m \leftarrow \mathcal{C}_m \cup \{x_j\}$;
end
return $\{\mathcal{C}_i,\ \text{LADC}(\mathcal{C}_i,\ \mathcal{A},\ k)\}_{i=1}^k$
```

2 Methods

2.1 Overview

We developed a new computational framework for the parallel *de novo* clustering analysis of ultra-large-scale sequence data, a task currently computationally intractable with conventional methods. The basic idea is to first partition data into small parts by using an incomplete hierarchical divisive tree, then process each part by using a user-chosen OTU picking method, and finally assemble individual clustering results to form the final output. Fig. 1 presents the flowchart of the proposed method, and the pseudo-code is given in Algorithm 1.

The development of the method is motivated by the following observation. Suppose that we have a dataset of N sequences, where N is on the order of $\mathcal{O}(10^9)$. We partition the data into M parts, and perform hierarchical clustering on each part using M processors. If each sub-dataset is of equal size, the overall computational complexity is $\mathcal{O}(N^2/M^2)$. If $M = 100$, theoretically, we could achieve 10^4-fold speed-up compared to the standard method. The above observation motivated us to develop a novel divide-and-conquer based approach. By partitioning data into small parts, we can significantly reduce the total number of sequence comparisons, and by feeding each sub-cluster into a computing node, the proposed method can be easily adapted to parallel computing environments. Our numerical experiments showed that if the parameters that control the height of a hierarchical divisive tree are properly set, the new method is able to achieve clustering accuracy comparable to the standard method but runs much faster even than heuristic methods.

2.2 Landmark-based Active Divisive Clustering

The core component of SLAD is the procedure that partitions data into small sub-clusters (Fig. 1), which has to meet two requirements. First, the partition process must be efficient; otherwise, the efficiency gain from parallelization is amortized. Second, the partition result must be accurate, since all the downstream operations depend on the top-level partition. We developed a new method, referred to as landmark-based active divisive clustering (LADC), that achieves the above two goals simultaneously. Below, we give a detailed discussion of the proposed method.

The LADC method partitions a sequence dataset recursively into clusters and represents them as an incomplete hierarchical divisive (HD) tree. An HD tree is a k-ary tree consisting of multiple layers of nodes with each node representing a cluster. It can be constructed by recursively partitioning a node into k children using a clustering method as one moves down the hierarchy. In a complete tree, each leaf node contains only one sequence. However, the partition process can stop intermediately so that each leaf node contains multiple sequences, thus forming an incomplete HD tree. The standard method for constructing an HD tree has a computational complexity of $\mathcal{O}(N^2)$, and hence is computationally infeasible to process large sequence datasets. One possible way to address the issue is to randomly select a small number of sequences and perform clustering analysis only on the selected sequences in each partition operation (Krishnamurthy *et al.*, 2012). In this way, the number of pairwise sequence comparisons can be significantly reduced. One issue associated with random selection is that samples in small clusters are seldom selected, and thus it may not be able to recover small clusters. In order to address the issue, we propose to construct an incomplete HD tree by using the adaptive landmark selection method (Voevodski *et al.*, 2012). The method was originally proposed for flat clustering, and to the best of our knowledge, it has never been used for constructing a data hierarchy.

The proposed method consists of three major steps. The first step is to select s landmark sequences from a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$. We start by randomly selecting a sequence x_{l_0} from the dataset that forms a landmark set $\mathcal{S}$. Then, we compute the pairwise distance between each sequence and the landmark set, randomly select a sequence from q sequences that are farthest from the landmark set, and put it into the landmark set. Here, the distance between a sequence and a landmark set is defined as the minimum distance between the sequence and a landmark sequence. The selection procedure is repeated until s landmark sequences are selected. Once we form a landmark set, the second step is to partition the landmark sequences into k clusters $\{\tilde{\mathcal{C}}_i\}_{i=1}^k$. For the purpose of this study, we used spectral clustering (Von Luxburg, 2007). Other clustering methods including k-means and k-medoids can also be used. However, one advantage of spectral clustering is that it is able to identify clusters of any shape, not merely limited to those with a hyper-sphere. The third step is to assign all non-landmark sequences $\{x_i : x_i \in \mathcal{D}\backslash\mathcal{S}\}$ to one of the k clusters $\{\tilde{\mathcal{C}}_i\}_{i=1}^k$. To this end, we compute the average distance between a sequence and the landmark sequences in each cluster, and assign it to the cluster with the minimal average distance. Since all the distances used in this step have already been computed in the first step, this step does not introduce any extra computational cost. The above three steps are iteratively performed on each cluster obtained in the previous partition until termination criteria are satisfied. The pseudo-code of LADC is presented in Algorithm 2.

2.3 Parameters

There are three parameters, namely k, q and s, that need to be determined in LADC. For ease of implementation, we set the number of division branches k to 2. In this way, an HD tree becomes a binary tree. It was suggested that q is set to be the average size of ground-truth clusters (Voevodski *et al.*, 2012). However, in our applications, the ground-truth clusters are generally unknown. A natural choice is to set $q = n/k$, where $k = 2$ is the number of clusters generated in each partition and n is the number of sequences in a cluster to be partitioned. Another important parameter is s, the number of landmark sequences selected. By following Voevodski *et al.* (2012), we set s to be $\log_2 n$. For the problems that we are most interested in, the number of sequences is on the order of $10^7 \sim 10^9$. Only $\sim$30 landmark sequences need to be selected. Thus, the selection of landmark sequences can be performed very efficiently with computational complexity of $\mathcal{O}(N \log_2 N)$.

We next discuss the termination criteria that we use to control the height of an HD tree, which is a trade-off between solution accuracy and computational efficiency. Three termination criteria are used, including the sub-cluster radius, the sub-cluster size, and the number of sub-clusters. Among them, the sub-cluster radius is the most important one. In order not to introduce extra computational costs, we estimate the radius of a node as the median of the pairwise distances between landmark sequences in the node. The reasoning is that if the result of spectral clustering performed on the landmark sequences is a good approximation of that obtained by spectral clustering performed on all sequences in a node, the estimated radius should be a good approximation of the radius of the node. Generally speaking, the probability of falsely separating sequences belonging to the same species increases as recursive bisection goes deeper. Hence, we can effectively control clustering accuracy by preventing clusters with small radiuses from being partitioned. In Section 4.2, we performed a parameter sensitivity analysis that demonstrated how to estimate a proper sub-cluster radius in order to achieve a good balance between solution accuracy and computational efficiency. Besides, we also use two auxiliary termination parameters, namely, the sub-cluster size and the number of sub-clusters. These two parameters are highly dependent on the input data size, so it is difficult to use them to control clustering quality. However, they can be used to force an early termination in order to balance the time spent on the top-level partition and sub-clustering phases.

2.4 Implementation

We implemented the proposed method on Apache Spark V2.0.2 by using the Scala programming language V2.11.8. Apache Spark is a fast and general engine for large-scale data processing, providing researchers with an interface for programming entire clusters with implicit data parallelism and fault-tolerance. It can run on Hadoop, Mesos, standalone, or in the cloud, and can access diverse data sources including HDFS, Cassandra, and HBase. Most existing parallel *de novo* OTU picking methods utilized message passing interface (MPI) for speed-up in a distributed computing environment (Matias Rodrigues and von Mering, 2013; Jiang *et al.*, 2016; Cai *et al.*, 2017). While MPI enables the message communication between computational nodes via network, it lacks job scheduling and fault recovery. Since our method can be easily fit into the MapReduce model, the low-level flexibility offered by MPI becomes less appealing. By using high-level and portable Apache Spark, our method is scalable, fault-tolerant, and compatible with different file systems. Apache Spark also supports several programming languages, including Python, R and Scala. We chose Scala since Apache Spark focuses on data transformation and mapping concepts, which are flawlessly supported by functional programming languages including Scala. Moreover, Scala is a JVM native language and thus is much more efficient than Python and R in Spark. Another advantage of using Apache Spark is that it is equipped with a bunch of built-in libraries. In our implementation, we used Spark MLlib (Meng *et al.*, 2016), which is a distributed framework built on top of Spark Core and provides a library of commonly used machine learning algorithms. Due in large part to the distributed memory-based Spark architecture, the implementations provided by Spark MLlib run much faster than disk-based counterparts. Due to space limitation, other implementation details are presented in Supplementary Data.

3 Theoretical Analysis

In this section, we present an analysis that provides theoretical guarantees for the proposed method on both accuracy and efficiency. We start by introducing some notations and definitions used in the analysis.

Definition 1. *A hierarchical clustering $\mathcal{T}$ on a dataset $\mathcal{D}$ is a collection of non-empty clusters that satisfy the following three constrains: 1) $\mathcal{C}_1 = \mathcal{D} \in \mathcal{T}$, 2) $\forall \mathcal{C}_i, \mathcal{C}_j \in \mathcal{T}$, either $\mathcal{C}_i \subset \mathcal{C}_j$, $\mathcal{C}_j \subset \mathcal{C}_i$ or $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, and 3) for any cluster $\mathcal{C} \in \mathcal{T}$, if $\exists \mathcal{C}'$ with $\mathcal{C}' \subset \mathcal{C}$, then there exist a set of disjoint clusters $\{\mathcal{C}_i\}_{i=1}^k$ so that $\bigcup_{i=1}^k \mathcal{C}_i = \mathcal{C}$.*

Each node in a hierarchical clustering $\mathcal{T}$ corresponds to a cluster. Specifically, the root node contains all the input sequences (constrain 1), any two nodes either have an ancestor-child relationship or have the same ancestor (constrain 2), and any non-terminal node has k child nodes (constrain 3). A hierarchy $\mathcal{T}$ can be constructed through a serial of k-ary splits (or partitions) in a top-down fashion. There are at most $N/(k-1)$ internal splits, where N is the number of sequences. Let $S_1, \cdots, S_m$ be the split of $\mathcal{C}_1, \cdots, \mathcal{C}_m$, respectively, where $m \leq N/(k-1)$. Each split has a parent split except for the root split, and each split has k child splits except for the leaf splits. We denote the parent split of S_i as $S_{\pi(i)}$, where $\pi(i)$ is the index of S_i's parent in the hierarchy.

In the proposed method, each internal split S_i consists of three phases: 1) adaptive landmark selection, 2) spectral clustering, and 3) averaging assignment. Denote as L_i, P_i and V_i the possible error events in the three phases, respectively. Our method fails if any of these error events occurs in an internal split. It is easy to see that $\mathbb{P}\{\text{failure}\} \leq \mathbb{P}\left\{\bigcup_{i=1}^m L_i \cup P_i \cup V_i\right\}$. Following the work of Krishnamurthy *et al.* (2012), the upper bound of the overall failure probability can be decomposed into the sum of probabilities of the three phases.

Lemma 1. *Let $B_1, B_2, \cdots, B_t$ be events in a measurable space. Then $\mathbb{P}\left\{\bigcup_{i=1}^t B_i\right\} \leq \sum_{i=1}^t \mathbb{P}\left\{B_i | \neg B_1, \cdots, \neg B_{i-1}\right\}$.*

Let B_i be an event in topological ordering $(\{L_1, P_1, V_1\}, \cdots, \{L_m, P_m, V_m\})$. By applying the following independence assertions: 1) each adaptive landmark selection phase is independent of previous error events and conditioned on the successful recovery of the corresponding parent clustering, 2) each spectral clustering phase is independent of previous failure events and conditioned on the success of landmark selection, and 3) each averaging assignment phase is independent of previous failures and conditioned on the success of landmark selection and spectral clustering, we have:

$$\begin{aligned}\mathbb{P}\{\text{failure}\} &\leq \mathbb{P}\{L_1\} + \mathbb{P}\{P_1|\neg L_1\} + \mathbb{P}\{V_1|\neg L_1, \neg P_1\} + \cdots \qquad (1)\\ &+ \mathbb{P}\{L_m|\neg L_1, \neg P_1, \neg V_1, \cdots, \neg L_{m-1}, \neg P_{m-1}, \neg V_{m-1}\}\\ &+ \mathbb{P}\{P_m|\neg L_1, \neg P_1, \neg V_1, \cdots, \neg L_{m-1}, \neg P_{m-1}, \neg V_{m-1}, L_m\}\\ &+ \mathbb{P}\{V_m|\neg L_1, \neg P_1, \neg V_1, \cdots, \neg L_{m-1}, \neg P_{m-1}, \neg V_{m-1}, L_m, P_m\}\\ &= \mathbb{P}\{L_1\} + \sum_{i=2}^m \mathbb{P}\{L_i|\neg P_{\pi(i)}, \neg V_{\pi(i)}\} + \sum_{i=1}^m \mathbb{P}\{P_i|\neg L_i\} + \sum_{i=1}^m \mathbb{P}\{V_i|\neg L_i, \neg P_i\}.\end{aligned}$$

After decomposition, we only need to consider the upper bound of the failure probability of each phase separately in the following analyses.

3.1 Adaptive Landmark Selection Phase

Definition 2. *An instance $(\mathcal{D}, d)$ satisfies the $(1+\alpha,\ \epsilon)$-property for the k-median objective function Φ with respect to the target clustering $\mathbb{C}^T$ if any clustering $\mathbb{C}$ with $\Phi(\mathbb{C}) \leq (1+\alpha)\Phi^*$ is ϵ-close to $\mathbb{C}^T$.*

In Definition 2, d is a distance function, $\mathcal{D} = \{x_i\}_{i=1}^N$ is the input data, and Φ^* is the optimal value of the objective function Φ. We say that two clusterings $\mathbb{C}$ and $\mathbb{C}'$ are ϵ-close if the fraction of points on which they disagree in terms of the optimal matching of these two clusters is at most ϵ. Let $\mathbb{C}^* = \{\mathcal{C}_j^*\}_{j=1}^k$ be the optimal k-median clustering. The jth sub-cluster and its center point are denoted by $\mathcal{C}_j^*$ and c_j^*, respectively. We define $w(x_i) = \min_{j=1}^k d(x_i, c_j^*)$ as the contribution of x_i to the objective function Φ. Hence, $\Phi^* = \sum_{i=1}^n w(x_i)$. We also define $w_2(x_i)$ as the distance between x_i and the second closest center point among $\{c_j^*\}_{j=1}^k$. Let us define the critical distance $d_{crit} = \alpha\bar{w}/17\epsilon$, where $\bar{w} = \Phi^*/n$ is the average weight. We say a point x_i is good if $w(x_i) < d_{crit}$ and $w_2(x_i) - w(x_i) \geq 17d_{crit}$; otherwise, x_i is bad. In addition, the set of good points can be partitioned into good sets $\{G_j\}_{j=1}^k$ so that $G_j \subset \mathcal{C}_j^*$. We can consider G_j as the core of cluster $\mathcal{C}_j^*$.

According to Voevodski *et al.* (2012) and Balcan *et al.* (2009), we have the following lemma:

Lemma 2. *Assume the optimal k-median clustering $\mathbb{C}^*$ satisfies the $(1+\alpha,\ \epsilon)$ property with respect to the target clustering $\mathbb{C}^T$, and each cluster in $\mathbb{C}^T$ has a size of at least $2\epsilon n$, then less than $(\epsilon - \epsilon^*)n$ points on which $\mathbb{C}^*$ and $\mathbb{C}^T$ agree have $w_2(x) - w(x) < \alpha\bar{w}/\epsilon$, and at most $17\epsilon n/\alpha$ points have $w(x) \geq \alpha\bar{w}/17\epsilon$.*

In Lemma 2, n is the number of input points and ϵ^* is the exact distance between $\mathbb{C}^*$ and $\mathbb{C}^T$. Thus, $\epsilon^* < \epsilon$. By the definition of bad point, the lemma bounds the number of bad points. We have at most $\epsilon^* n + (\epsilon - \epsilon^*)n + 17\epsilon n/\alpha = (1 + 17/\alpha)\epsilon n = b$ bad points.

Definition 3. *A landmark set $\mathcal{S}$ satisfies the landmark spread property if for any G_i there exists a landmark in $\mathcal{S}$ with a distance smaller than $2d_{crit}$ to a certain point in G_i.*

Lemma 3. *Given the number of clusters k and some $\delta > 0$, let $s = 4k + 16\ln(1/\delta)$ and $q = 2b$. Assume that an instance $(\mathcal{D}, d)$ satisfies the $(1+\alpha,\ \epsilon)$-property for the k-median objective function and each cluster in the target clustering $\mathbb{C}^T$ has a size of at least $(4 + 51/\alpha)\epsilon n$. With probability at least $1-\delta$, the landmark set returned in Algorithm 2 satisfies the landmark spread property.*

Proof. By Lemma 2, $\mathbb{C}^*$ is ϵ-close to $\mathbb{C}^T$, and there are at most b bad points. Since each cluster in the target clustering has at least $(4+51/\alpha)\epsilon n$ points, we have $|G_i| \geq (4+51/\alpha)\epsilon n - \epsilon n - b = (2 + 34/\alpha)\epsilon n = 2b$, which means each good set has at least $2b$ good points.

We define a random variable I_i as an indicator of choosing a good point at the ith iteration so as to bound the probability of selecting less than k good points. A good point is selected at the ith iteration if $I_i = 1$; otherwise, $I_i = 0$. Random variables $\{I_i\}$ are independent and identically distributed. For s iterations, the number of selected good points is $\bar{I} = \sum_{i=1}^s I_i$. Since there are at most b bad points, the probability of uniformly selecting a good point from $2b$ points is $\mathbb{P}\{I_i\} \geq 1/2$. The expectation of selecting a good point is $\mu = \mathbb{E}[\bar{I}] = \sum_{i=1}^s \mathbb{E}[I_i] = \sum_{i=1}^s (0 \times \mathbb{P}\{I_i = 0\} + \mathbb{P}\{I_i = 1\}) \geq s/2$. By the Chernoff bound, we have $\mathbb{P}\{\bar{I} < (1-\delta)\mu\} \leq e^{-\mu\delta^2/2}$, where $0 < \delta < 1$. If $s = 4k + 16\ln(1/\delta) > 2k$, we have $\mathbb{P}\{\bar{I} < k\} \leq e^{-\frac{s}{4}(1-\frac{2k}{s})^2} \leq e^{-(4k+16\ln(1/\delta))/16} \leq \delta$. Therefore, the probability of selecting less than k good points is smaller than δ after s iterations.

Once we have selected k good points, we need to prove that they satisfy the landmark spread property. There are two possible cases. In case 1, good points are selected from distinct good sets. The landmark spread property trivially holds. In case 2, at least two good points are selected from the same good set. Suppose that x_i and x_j are two good points from the same good set. Let $\mathcal{S}$ be the landmark set at the moment and $d(x, \mathcal{S}) = \min_{l \in \mathcal{S}} d(x, l)$ be the distance between x and point set $\mathcal{S}$. Without loss of generality, we assume that x_j is selected after x_i. According to the triangle inequality implied by the metric space assumption, $d(x_j, \mathcal{S}) \leq d(x_i, x_j) < 2d_{crit}$. Moreover, x_j is chosen from the farthest $q = 2b$ points. Therefore, when x_j is chosen, at least $n - 2b + 1$ points x satisfy $d(x, \mathcal{S}) \leq d(x_j, \mathcal{S}) < 2d_{crit}$. Hence, there must exist a landmark with distance closer than $2d_{crit}$ to a certain point in each good set. □

3.2 Spectral Clustering Phase

Lemma 4. *If a landmark set $\mathcal{S}$ satisfies the landmark spread property over $\{G_j\}_{j=1}^k$, $d(x_l, x_{l'})$ is either larger than $12d_{crit}$ or smaller than $6d_{crit}$ for any $x_l \in \mathcal{S}$ and $x_{l'} \in \mathcal{S}$.*

Proof. Let x_l be a landmark that satisfies $d(x_l, x_i) < 2d_{crit}$ for a good point $x_i \in G_j$. For any $x_p \in G_j$, we have $d(x_p, x_l) \leq d(x_p, x_i) + d(x_i, x_l) < 4d_{crit}$. For any $x_{p'} \in G_{j'}, j' \neq j$, we have $d(x_{p'}, x_l) \geq d(x_{p'}, x_i) - d(x_i, x_l) > 16d_{crit} - 2d_{crit} = 14d_{crit}$. In case 1, we assume $d(x_l, x_i) < 2d_{crit}$, $d(x_{l'}, x_{i'}) < 2d_{crit}$, and $i = i'$ for landmarks $x_l \in \mathcal{S}$, $x_{l'} \in \mathcal{S}$ and good points $x_i \in G_i$ and $x_{i'} \in G_{i'}$. Then, we have $d(x_l, x_{l'}) \leq d(x_l, x_i) + d(x_i, x_{l'}) < 2d_{crit} + 4d_{crit} = 6d_{crit}$. In case 2, we assume $d(x_l, x_i) < 2d_{crit}$, $d(x_{l'}, x_{i'}) < 2d_{crit}$, and $i \neq i'$ for landmarks $x_l \in \mathcal{S}$, $x_{l'} \in \mathcal{S}$ and good points $x_i \in G_i$ and $x_{i'} \in G_{i'}$. Then, we have $d(x_l, x_{l'}) \geq d(x_l, x_{i'}) - d(x_{i'}, x_{l'}) > 14d_{crit} - 2d_{crit} = 12d_{crit}$. □

Lemma 5. *Spectral clustering can obtain a clustering over a landmark set, where landmarks whose nearest good points belonging to the same good set are grouped to the same cluster, and landmarks whose nearest good points belonging to different good sets are assigned to different clusters.*

Proof. By Lemma 3, given a landmark set $\mathcal{S} = \{x_{l_i}\}_{i=1}^s$, each x_{l_i} must be closer than $2d_{crit}$ to a certain point in a good set $G_{\varphi(l_i)}$, where φ is a mapping from l_i to the index of its closest good set. Let $\{\mathcal{S}_i\}_{i=1}^k$ be the partition result of landmark set $\mathcal{S}$. Lemma 4 states that the distance is smaller than $6d_{crit}$ if two landmarks are assigned to the same cluster, and larger than $12d_{crit}$ otherwise. Let K be a similarity function. Spectral clustering solves the following optimization problem to obtain the optimal clustering: $\min_{\{\mathcal{S}_i\}_{i=1}^k} \sum_{i=1}^k \frac{1}{|\mathcal{S}_i|} \sum_{x_{l_p} \in \mathcal{S}_i, x_{l_q} \in \mathcal{S} \setminus \mathcal{S}_i} K(x_{l_p}, x_{l_q})$. The intuition is to separate points in different groups according to their similarities: the similarity of two points in the same group is high, while the similarity of two points in different groups is low. This is obvious for the landmark set $\mathcal{S}$ according to Lemma 4. Hence, the points located far away from each other ($> 12d_{crit}$) are assigned to different clusters. For $k = 2$, the optimization problem is exactly the unnormalized spectral clustering problem, and for $k > 2$, k-means method is usually applied on the projected space (Von Luxburg, 2007). □

3.3 Averaging Assignment Phase

We define the average distance between point x and point set $\mathcal{C} = \{x_1, \cdots, x_m\}$ as $d(x, \mathcal{C}) = \frac{1}{m}\sum_{i=1}^m d(x, x_i)$. The following lemma states that any point that is not in the good set but satisfies $w_2(x) - w(x) \geq 17d_{crit}$ can be assigned correctly in the averaging assignment phase.

Lemma 6. *Let $\{\mathcal{S}_i, \cdots, \mathcal{S}_k\}$ be the partition result returned by spectral clustering on the selected landmark set $\mathcal{S}$. For any good point x in $\mathcal{C}_i^*$, we have $d(x, \mathcal{S}_i) < d(x, \mathcal{S}_{i'})$ if $i' \neq i$ and $\mathcal{S}_i \subseteq \mathcal{C}_i$.*

Proof. Let c_i be the center of cluster $\mathcal{C}_i$. By the definition of good point, we have $c_i \in G_i$. The average distance between c_i and landmark set $\mathcal{S}_i = \{x_{i_1}, \cdots, x_{i_m}\}$ is $d(c_i, \mathcal{S}_i) < 5d_{crit}$. To see this, let x be a good point that is grouped by spectral clustering into a cluster containing G_i. Hence, $d(x, x_{i_j}) < 4d_{crit}$ based on the proof of Lemma 4, and $d(x, c_i) = w(x) < d_{crit}$. Thus, we have $d(c_i, x_{i_j}) \leq d(x, x_{i_j}) + d(x, c_i) < 5d_{crit}$ for $x_{i_j} \in \mathcal{S}_j$. It follows that $d(c_i, \mathcal{S}_i) = \frac{1}{m}\sum_{j=1}^m d(c_i, x_{i_j}) <$

Table 1. Averaged running time (in second) of four methods performed on the plaque and Greengenes (GG) datasets with and without SLAD. For the Greengenes dataset, only UCLUST finished the analysis in 72-hour wall-time limit. When a method was used with SLAD, the total running time is the sum of the time spent on top-level partition and sub-clustering. The experiment was performed on a 4×2.4GHz Intel Xeon E5645 processor machine.

Data	Method	w/o SLAD	with SLAD: top-level partition	with SLAD: sub-clustering	Speed-up
Plaque	UCLUST	507	160	130	1.8
	Cd-hit	4344	160	691	5.1
	AbundantOTU	41391	160	4622	8.7
	ESPRIT-Tree	12067	160	5017	2.3
GG	UCLUST	17247	1325	2101	5.0

$5d_{crit}$. By triangle inequality, we have the following results:

$$d(x,\mathcal{S}_i) = \frac{1}{|\mathcal{S}_i|}\sum_{j=1}^{|\mathcal{S}_i|} d(x, x_{i_j}) \le \frac{1}{|\mathcal{S}_i|}\sum_{j=1}^{|\mathcal{S}_i|}\Big(d(x,c_i) + d(c_i, x_{i_j})\Big)$$

$$< w(x) + \frac{1}{|\mathcal{S}_i|}\sum_{j=1}^{|\mathcal{S}_i|} d(c_i, x_{i_j}) = w(x) + 5d_{crit}\ ,$$

$$d(x,\mathcal{S}_{i'}) = \frac{1}{|\mathcal{S}_{i'}|}\sum_{j=1}^{|\mathcal{S}_i|} d(x, x_{i'_j}) \ge \frac{1}{|\mathcal{S}_{i'}|}\sum_{j=1}^{|\mathcal{S}_{i'}|}\Big(d(x,c_{i'}) + d(c_{i'}, x_{i'_j})\Big)$$

$$> w_2(x) - \frac{1}{|\mathcal{S}_{i'}|}\sum_{j=1}^{|\mathcal{S}_{i'}|} d(c_{i'}, x_{i'_j}) > w(x) + 12d_{crit}\ .$$

Thus, $d(x,\mathcal{S}_i) < w(x) + 5d_{crit} < w(x) + 12d_{crit} < d(x,\mathcal{S}_{i'})$. □

After the spectral clustering and averaging assignment phases, all good points are correctly clustered. Since there are at most b bad points, the distance between clustering $\mathbb{C}$, which is generated by Algorithm 2, and $\mathbb{C}^*$ is at most $\mathcal{O}(\epsilon/\alpha)$. Thus, $\mathbb{C}$ is at least $\mathcal{O}(\epsilon/\alpha + \epsilon)$-close to $\mathbb{C}^T$.

3.4 Main Theoretical Results

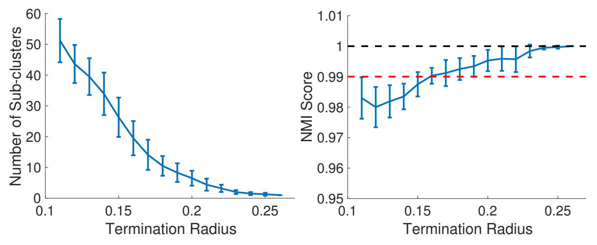
To sum up, we present our final main theoretical results.

Theorem 1. *Let $\mathcal{D}$ be a dataset with a hierarchy $\mathcal{T}^*$. Assume that an instance $(\mathcal{D}, d)$ in some metric space satisfies the $(1+\alpha,\epsilon)$-property for the k-median objective function and each split S_i in $\mathcal{T}^*$ has a size of at least $(4 + 51/\alpha)\epsilon|\mathcal{C}_i|$. The following results hold for Algorithm 2: 1) A hierarchy ϵ-close to the true hierarchy can be obtained with probability $1-\mathcal{O}(1)$, if the number of landmarks $s \ge 4k + 16\ln\frac{N}{k-1}$, and 2) the total number of distance calculations is $\mathcal{O}(sN\log N)$.*

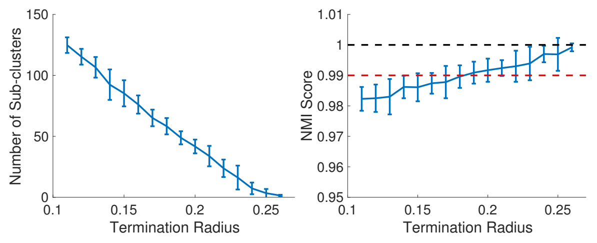
Proof. By Lemmas 3, 5, 6, and inequality (1), we have $\mathbb{P}\{\text{failure}\} \le \frac{N}{k-1}\delta$. By Lemma 3, $s = 4k + 16\ln(1/\delta)$. In order to achieve $\mathcal{O}(1)$, let $\delta = \frac{k-1}{N}$. It follows that $s \ge 4k + 16\ln\frac{N}{k-1}$. In each split S_i, we need to calculate the distances to all selected landmarks for each point in $\mathcal{C}_i$, and the splitting tree has $\log N$ levels. Thus, the total number of distance calculations involved is $\mathcal{O}(Ns\log N)$. □

4 Results

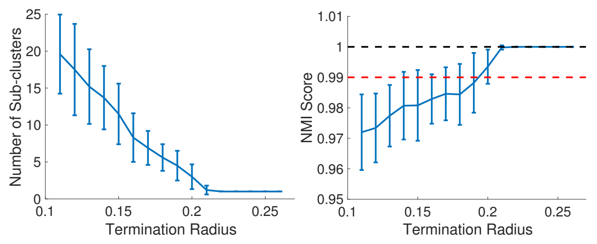
We performed a large-scale experiment to demonstrate that the proposed framework can significantly speed up various commonly used methods for *de novo* OTU picking and meanwhile maintain the same level of accuracy.



(a) plaque data



(b) Greengenes data



(c) EMP-755

Fig. 2. Parameter sensitivity analysis performed on (a) plaque, (b) Greengenes, and (c) EMP-755 datasets. The first and second columns report the numbers of sub-clusters and NMI scores obtained after the top-level partition by using different termination radiuses and the subsequent mock sub-clustering, respectively. The radius thresholds for the three datasets were estimated to be 0.17, 0.19, and 0.20, respectively.

4.1 Datasets

When evaluating an OTU picking method for sequence analysis, clustering accuracy and computational efficiency are two major considerations. Accordingly, four datasets were used in the experiment. The first dataset was generated from oral plaque samples that cover the V3-V4 hyper-variable regions of 16S rRNA gene. To generate species-level taxonomic labels for the dataset, we performed BLAST search against the HOMD database (Chen *et al.*, 2010) and annotated each sequence by using a stringent criterion: the identity percentage ≥97% and the length of the aligned region ≥97% of the total length. A total of 410,600 sequences were confidently annotated at the species level. The second dataset is the Greengenes database (McDonald *et al.*, 2012), which is one of the most commonly used databases for 16S rRNA gene sequence annotation and contains 1,269,986 taxonomically labeled sequences spanning over the V1-V9 hyper-variable regions. The third dataset, which contains 66,520,485 sequences of the V4 region, comes from a study of a water purification system (Haig *et al.*, 2014). Since it is one of the studies performed in the Earth Microbiome Project (EMP) (Gilbert *et al.*, 2014) (study #755), we refer to it as the EMP-755 dataset. The fourth dataset is the whole EMP dataset, consisting of 27,751 samples from 97 studies. The dataset has ∼2.2 billion V4 16S rRNA sequences and is probably the largest publicly available 16S rRNA sequence dataset.

4.2 Parameter Sensitivity Analysis

In the proposed method, the termination of top-level partition plays a critical role in determining the trade-off between clustering quality and computational efficiency. We proposed to use the sub-cluster radius as a termination criterion. Here, we performed a parameter sensitivity analysis to demonstrate that the proposed method suffers a minimal loss in clustering accuracy when the termination parameter is properly set. Three datasets were used, namely, plaque (V3-V4), Greengenes (V1-V9) and EMP-755 (V4). The first two datasets have already been annotated. Since it is computationally expensive to annotate the entire EMP-755 dataset, we randomly sampled 1M sequences and annotated the sequences by searching against the Greengenes database using USEARCH (Edgar, 2010). Given an annotated dataset, we randomly extracted 80% sequences without replacement, applied LADC to the extracted sequences by using different termination radiuses ranging from 0.11 to 0.26, and repeated the above process 10 times. Since LADC can be used with various *de novo* OTU picking method, it is likely that a termination threshold is dependent on the clustering method used in the subsequent sub-clustering phase. In order to derive a generally applicable termination threshold and assess the performance loss incurred from using LADC, we assumed that the sub-clustering phase is perfect and mocked it so that as long as sequences with the same taxonomic label are not falsely partitioned into different clusters at the top level, they can always be correctly grouped at the sub-clustering phase. After the top-level partition and mock sub-clustering, we calculated a NMI (normalized mutual information) score by comparing the result with known sequence annotations.

Fig. 2 reports the numbers of sub-clusters and NMI scores obtained after the top-level partition using different termination radiuses. As expected, the numbers of sub-clusters decrease and the NMI scores increase with respect to the termination radius. To select a proper termination threshold, we performed a one-side paired t-test at each radius level and picked the smallest radius level that accepted the alternative hypothesis that the NMI score loss is significantly smaller than 0.01 (P-value < 0.05). Note that the selected termination thresholds are slightly different for sequences covering different hyper-variable regions. It is also worth pointing out that an NMI score loss of 0.01 is very small. As we will see shortly, the application of different *de novo* OTU picking methods to the same dataset can have 0.05 difference in NMI scores (see Fig. 3).

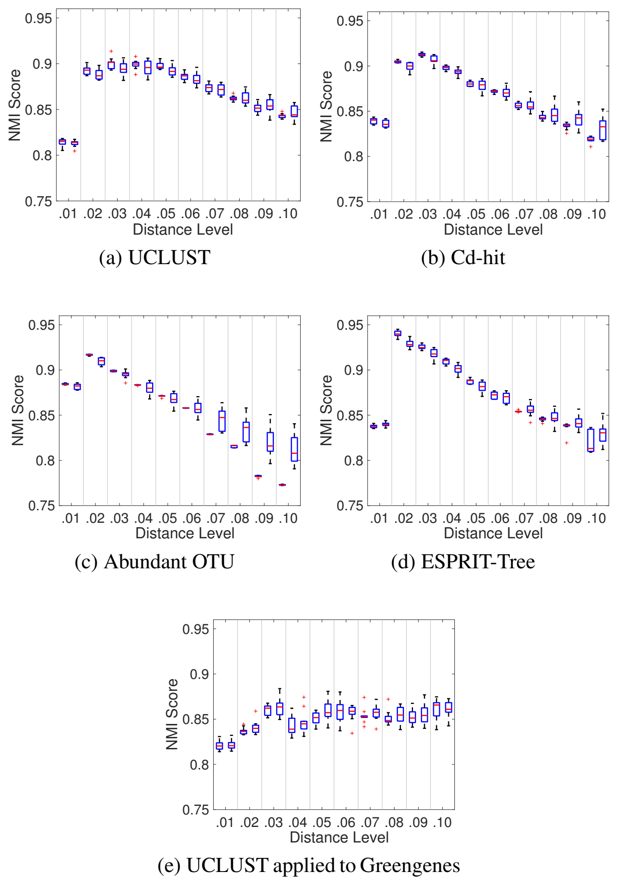


Fig. 3. Averaged NMI scores obtained at the ten distance levels for four tested methods performed on (a-d) the plaque and (e) Greengenes datasets. At a given distance level, the first box plot shows the NMI scores obtained without SLAD, and the second one shows NMI scores obtained with SLAD applied.

4.3 Benchmark Study on Clustering Quality

The proposed method can in principle be used to parallelize *any* single-thread *de novo* OTU picking method. To demonstrate this, we applied four different methods to the plaque and Greengenes datasets. UCLUST V9.0 (Edgar, 2010) and Cd-hit V4.6 (Li and Godzik, 2006) are two most commonly used heuristic methods. AbundantOTU V0.93 (Ye, 2010) is a consensus alignment based method. ESPRIT-Tree (Cai and Sun, 2011) is a fast implementation of hierarchical clustering method. For a given dataset, we first randomly sampled 80% sequences, grouped the sampled sequences at various distance levels ranging from 0.01 to 0.10, and compared NMI scores obtained with and without SLAD. To minimize statistical variations, the above process was repeated 10 times. The termination radius parameter used in the top-level partition was set to 0.17 for the plaque dataset and 0.19 for the Greengenes dataset as determined above. The experiment was performed on a 4×2.40GHz Intel Xeon E5645 processor machine.

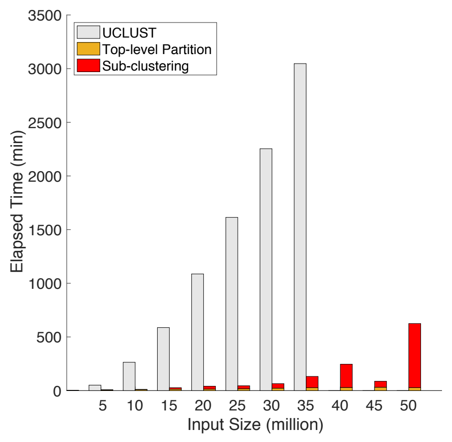
Fig. 3 reports the averaged NMI scores evaluated at the ten distance levels for the two datasets. For the experiments performed on the Greengenes dataset, only UCLUST finished in 72 hours, which is the wall-time limit of our computing cluster, so only UCLUST results are presented. For each tested method at a given distance level, the first and second box plots show the NMI scores obtained without and with SLAD applied, respectively. We used a one-side paired t-test to compare two sets of NMI scores. With only one exception (ESPRIT-Tree at the 0.02 distance level), all tests accepted the alternative hypothesis that the NMI score loss is significantly smaller than 0.01 at P-value < 0.05. This is consistent with the results shown in the parameter sensitivity analysis. We noted that at some distance levels, the NMI scores obtained with SLAD can be even larger than those obtained without SLAD. This can be explained by the fact that OTU picking methods used in the sub-clustering phase is not perfect as we assumed in the parameter sensitivity analysis, and when the top-level partition correctly separates sequences with different taxonomic labels, it prevents from a possible false merge at the sub-clustering phase. Following one of the reviewers' suggestion, we also computed the NMI scores by comparing the clustering results obtained by the four tested methods with and without SLAD and reported the results in Supplementary Fig. 1. At the 0.03 and 0.05 distance levels (the two commonly used thresholds for defining species- and genus-level OTUs, respectively (Caporaso *et al.*, 2010)), the NMI scores stay at a very high level (0.97 $\sim$ 0.99) across all datasets and all tested methods.

Table 1 reports the average running time of the four tested methods with and without SLAD. In general, a clustering method can utilize only a single core, but when SLAD is applied to generate sub-clusters, all 4 cores can be used. Notably, the speed-up can go beyond 4-fold, which is the maximum speed-up that one can achieve through naive parallel computing. This is because the generation of sub-clusters at the top level reduces the search space for subsequent sub-clustering, which further boosts the computational efficiency. We should point out that SLAD is designed for large-scale sequence clustering analysis, and we will shortly observe even more significant speed-up when it is applied to the EMP dataset.

4.4 Scalability Study

We finally conducted a large-scale scalability study on the EMP-755 and entire EMP datasets. For computational considerations, only UCLUST was tested. To investigate how the running time of UCLUST with and without SLAD grows with respect to the number of input sequences, we randomly sampled various numbers of sequences (5M, 10M, 15M, 20M, 25M, 30M, 35M, 40M, 45M, 50M) from the EMP-755 dataset. The termination radius parameter used in the top-level partition in SLAD was set to 0.20 as per the parameter sensitivity analysis, and the distance-level parameter of UCLUST was set to 0.03. The experiment was performed on a 4×2.40GHz Intel Xeon E5645 processor machine. Fig. 4 reports the running time of UCLUST with and without SLAD. Since UCLUST applied to 40M, 45M, and 50M sequences did not finish in 72 hours, the results are not reported. With only one exception (5M), SLAD accelerated UCLUST by more than one order of magnitude. Also note that the running time of UCLUST with SLAD grows much more slowly than that without SLAD with respect to the input data size. This suggests that the proposed method has the potential to achieve even more speed-up on larger datasets, as shown below. We also compared the clustering results obtained by UCLUST with and without SLAD and the NMI scores are around $0.97 \sim 0.98$ (Fig. 4), which is consistent with the result observed in Supplementary Fig. 1.

To further demonstrate the scalability of the proposed method, we conducted an experiment on the entire EMP dataset. To our knowledge, this is the largest *de novo* 16S rRNA sequence clustering analysis ever performed in a distributed computing environment. We first transferred the data to Amazon Web Server (AWS) S3 and requested a computing cluster consisting of 17 m3.xlarge (Intel Xeon E5-2680 V2 Ivy Bridge Processors, 4 cores, 15GB memory) Amazon Elastic Compute Cloud (Amazon EC2) instances. The Apache Spark computing environment was then set up via AWS Elastic Map-reduce service (EMR) V5.6.0. The cluster was launched in a client mode, where 16 slave instances were used for computation and a master node was used for monitoring. The memory limit was set to 10,473MB for the master node and 9,658MB for the slave nodes. The termination radius parameter and the distance-level parameter of UCLUST were the same as above. We also set the number of sub-clusters to 300 for an early termination. The top-level partition phase took 533 minutes and the sub-clustering phase took 536 minutes. The total running time was $\sim$17.8 hours. In contrast, it has been estimated that the running time of UCLUST applied to a subset of the EMP dataset that contains $\sim$660M sequences is 150 days on a single computer (Rideout *et al.*, 2014). We have previously shown that the empirical computational complexity of UCLUST is $\mathcal{O}(N^{1.2}) \sim \mathcal{O}(N^{1.3})$ (Sun *et al.*, 2011). Thus, if UCLUST were applied to the entire EMP data, it would take $\sim$636 days.



Input Size	5M	10M	15M	20M	25M	30M	35M	40M	45M	50M
Speed-up	6	24	22	27	36	35	23	N/A	N/A	N/A
NMI Score	0.98	0.97	0.97	0.97	0.97	0.98	0.97	N/A	N/A	N/A

Fig. 4. Results of UCLUST with and without SLAD performed on various numbers of sequences sampled from the EMP-755 dataset. When UCLUST was used with SLAD, the running time is the sum of the time spent on top-level partition and sub-clustering. The NMI scores compare the clustering results obtained by UCLUST with and without SLAD. UCLUST did not finish in 72 hours when it was applied to 40M, 45M, and 50M sequences, so the results were not reported.

5 Conclusion

In this paper, we have developed a novel two-stage parallel sequence clustering framework that addresses the computational issue of existing methods for ultra-large-scale sequence analysis. Theoretical results have showed that our method can recover the true hierarchy with a high probability under mild assumptions and has a theoretical linearithmic time complexity with respect to the number of input sequences. In addition, we have demonstrated that the proposed method can efficiently process ultra-large-scale sequence datasets by taking advantage of parallel computing resources with the implementation on Apache Spark.

Funding

This work is in part supported by 1R01AI125982 (YS, RG, JWW), 1R01DE024523 (JWW, RG, YS, MB, WZ), and National Science Foundation of China (YC, grant # 11471313).

References

Balcan,M.F. *et al.* (2009) Approximate clustering without the approximation. *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1068-1077.
Cai,Y. and Sun,Y. (2011) ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. *Nucleic Acids Res.*, **39**, e95.
Cai,Y. *et al.* (2017) ESPRIT-Forest: Parallel clustering of massive amplicon sequence data in subquadratic time. *PLOS Comput. Biol.*, **13**, e1005518.
Caporaso,J.G. *et al.* (2010) QIIME allows analysis of high-throughput community sequencing data. *Nat. Methods*, **7**, 335-336.
Chen,T. *et al.* (2010) The Human Oral Microbiome Database: a web accessible resource for investigating oral microbe taxonomic and genomic information. *Database*, **2010**, baq013.
Chen,W. *et al.* (2013) MSClust: a multi-seeds based clustering algorithm for microbiome profiling using 16S rRNA sequence. *J. Microbiol. Methods*, **94**, 347-355.
Di Bella,J.M *et al.* (2013) High throughput sequencing methods and analysis for microbiome research. *J. Microbiol. Methods*, **95**, 401-414.
Edgar,R.C. (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, **26**, 2460-2461.
Editorial (2013) Your microbes, your health. *Science*, **342**, 1440-1441.
Gilbert,J.A. *et al.* (2014) The Earth Microbiome project: successes and aspirations. *BMC Biol.*, **12**, 69.
Haig,S.J. *et al.* (2014) Replicating the microbial community and water quality performance of full-scale slow sand filters in laboratory-scale filters. *Water Res.*, **61**, 141-151.
Jiang,L. *et al.* (2016) DACE: a scalable DP-means algorithm for clustering extremely large sequence data. *Bioinformatics*, **33**, 834-842.
Krishnamurthy,A. *et al.* (2012) Efficient active algorithms for hierarchical clustering. *Proc. 29th International Conference on Machine Learning*, 887-894.
Li,W. and Godzik, A. *et al.* (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, **22**, 1658-1659.
Mande,S.S. *et al.* (2012) Classification of metagenomic sequences: methods and challenges. *Brief. Bioinform.*, **13**, 669-681.

McDonald,D. *et al*. (2012) An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *ISME J.*, **6**, 610.

Meng,X. *et al*. (2016) MLlib: Machine learning in Apache Spark. *J. Mach. Learn. Res.*, **17**, 1235-1241.

Rideout,J.R. *et al*. (2014) Subsampled open-reference clustering creates consistent, comprehensive OTU definitions and scales to billions of sequences. *PeerJ*, **2**, e545.

Matias Rodrigues, J.F. and von Mering, C. (2013) HPC-CLUST: distributed hierarchical clustering for large sets of nucleotide sequences. *Bioinformatics*, **30**, 287-288.

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

Schloss,P.D. and Westcott,S.L. (2011) Assessing and improving methods used in operational taxonomic unit-based approaches for 16S rRNA gene sequence analysis. *Appl. Environ. Microbiol.*, **77**, 3219-3226.

Slaney,M. and Casey,M. (2008) Locality-sensitive hashing for finding nearest neighbors. *IEEE Signal Process. Mag.*, **25**, 128-131.

Sun,Y. *et al*. (2011) A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis. *Brief. Bioinform.*, **13**, 107-121.

Sun,Y. *et al*. (2009) ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Res.*, **37**, e76.

Sun,Y. *et al*. (2010) Advanced computational algorithms for microbial community analysis using massive 16S rRNA sequence data. *Nucleic Acids Res.*, **38**, e205.

Voevodski,K. *et al*. (2012) Active clustering of biological sequences. *J. Mach. Learn. Res.*, **13**, 203-225.

Von Luxburg, U. (2007) A tutorial on spectral clustering. *Stat. Comput.*, **17**, 395-416.

Ye,Y. (2010) Identification and quantification of abundant species from pyrosequences of 16S rRNA by consensus alignment. *Proc. 2010 IEEE International Conference on Bioinfomatics and Biomedicine*, 153-157.

A Parallel Computational Framework for Ultra-large-scale Sequence Clustering Analysis

Supplementary Data

Wei Zheng, Qi Mao, Robert J. Genco, Jean Wactawski-Wende
Michael Buck, Yunpeng Cai, Yijun Sun*

1 Implementation Details

1.1 Apache Spark

We implemented the proposed method on Apache Spark V2.0.2 by using the Scala programming language V2.11.8. Apache Spark is a fast and general engine for large-scale data processing, which provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance. It can run on Hadoop, Mesos, standalone, or in the cloud, and can access diverse data sources including HDFS, Cassandra, HBase and S3. Most existing parallel *de novo* OTU picking methods utilized message passing interface (MPI) for speed-up in a distributed computing environment [1, 5, 8]. While MPI enables the message communication between computational nodes via network, it lacks job scheduling and fault recovery. Since our method can be easily fit into the MapReduce model, the low-level flexibility offered by MPI becomes less appealing. By using high-level and portable Apache Spark, our method is scalable, fault-tolerant, and compatible with different file systems. In addition, Apache Spark supports several programming languages, including Python, R and Scala. We chose Scala since Apache Spark focuses on data transformation and mapping concepts, which are flawlessly supported by functional programming languages including Scala. Moreover, Scala is a JVM native language and thus is much more efficient than Python and R in Spark.

Apache Spark also provides users with a programming interface centered on a data structure called resilient distributed dataset (RDD), a read-only multi-set of data items distributed over a cluster of machines and maintained in a fault-tolerant way. It addresses the limitation of the MapReduce cluster computing paradigm, which always forces a program to read input data from disk. In our method, landmarks are selected in an iterative fashion. The frequent access to the data stored in memory rather than disk can save a huge amount of computational time by avoiding unnecessary I/O operations.

*Please address all correspondence to Dr. Yijun Sun (yijunsun@buffalo.edu)

1.2 Duplication Removal and Abundance Filtering

A sequence may appear multiple times in an input sequence collection. Identifying duplicated sequences is a necessary pre-processing step to reduce unnecessary distance calculations. After duplication removal, we kept only unique sequences and their abundance information, which records the frequencies of unique sequences in raw data. Abundance filtering that removes sequences with abundance lower than a pre-defined threshold can further speed up the OTU picking process. By default, we set the threshold parameter to 2, which is equivalent to singleton removal. Since the sequence abundances have a power-law distribution [9], sequences that pass abundance filtering account for the overwhelming majority of all input sequences. To avoid potential clustering accuracy loss, we only applied abundance filtering in the landmark selection step. At the final assignment step, all sequences were clustered based on their average distances to selected landmarks.

1.3 Distance Calculation

Distance calculation is a core component in OTU picking analysis. It determines clustering structure to be detected and consumes the majority of the computation time. The optimal method to measure pairwise sequence distances is through sequence alignment [10]. However, sequence alignment usually exhibits quadratic computational complexity with respect to the sequence length, and thus is computationally intractable to process large sequence datasets. To reduce computational complexity, the k-mer distance shown below is widely used to approximate the alignment distance [2, 4]:

$$d(x, y) = 1 - \sum_{i=1}^{|\mathcal{K}|} \min(c_x(i), c_y(i))/(\min(l_x, l_y) - k + 1) \ . \tag{1}$$

Given two sequences x, y and a pre-defined parameter k, each sequence is converted to a set of k-mers by sliding a window of size k over the entire sequence. Denote as $\mathcal{K}$ the union of the k-mer sets generated from the two sequences, $c_x(i)$ and $c_y(i)$ as the occurrences of the ith k-mer in sequences x and y, respectively, and l_x and l_y as the lengths of sequences x and y, respectively. The k-mer distance is essentially the fraction of non-shared k-mers between two sequences, and can be calculated in linear time with respect to the sequence length. Note that every sequence has to be converted into a list of k-mer counting numbers before distance calculation. We thus stored the RDD of k-mer counting lists instead of that of raw sequences in the memory. In this way, we can save a huge amount of time spent on data conversion at the cost of extra memory consumption. However, since landmarks are selected only from abundant sequences, we can do so in the landmark selection phase without running out of memory.

1.4 Median of Medians

In every landmark selection loop, we need to select a landmark from the farthest $n/2$ points, where n is the number of sequences in a cluster to be partitioned. A trivial solution is to sort all non-landmark sequences based on their distances to a landmark set, and then randomly select a sequence from the farthest $n/2$ sequences. However, global sorting is a very expensive operation in a distributed computing environment. Inspired by the median of medians algorithm [3], which finds an approximate median in linear time, we used the median of medians as a pivot sequence. Specifically, in Apache Spark, the data within an RDD is split into several partitions. A partition never spans multiple machines, and the default number of partitions equals to the total number of cores on all executor nodes. To avoid global sorting, we performed sorting on each partition, collected their medians, and used the median of collected medians as a pivot sequence to select a landmark.

1.5 Spark MLlib

Another advantage of using Apache Spark is that it is equipped with a bunch of built-in libraries, which can significantly simplify the construction of large-scale computational pipelines. For this study, we used Spark MLlib [7], which is a distributed framework built on top of Spark Core and provides a library of commonly used machine learning and statistical algorithms. Due in large part to the distributed memory-based Spark architecture, the implementations provided by Spark MLlib run much faster than disk-based counterparts. In our method, we used Power Iteration Clustering (PIC) to perform spectral clustering on selected landmarks. PIC [6] is a scalable and efficient algorithm for clustering vertices of a graph given pairwise similarities as edge properties. It computes the pseudo-eigenvectors of the normalized affinity matrix of a graph via power iteration, and has a much lower computational complexity compared to other methods for similar tasks.

References

[1] Y. Cai, W. Zheng, J. Yao, Y. Yang, V. Mai, Q. Mao, and Y. Sun. ESPRIT-Forest: Parallel clustering of massive amplicon sequence data in subquadratic time. *PLoS Computational Biology*, 13(4):e1005518, 2017.

[2] J. G. Caporaso, J. Kuczynski, J. Stombaugh, K. Bittinger, F. D. Bushman, E. K. Costello, N. Fierer, A. G. Peña, J. K. Goodrich, J. I. Gordon, et al. QIIME allows analysis of high-throughput community sequencing data. *Nature Methods*, 7(5):335–336, 2010.

[3] T. H. Cormen. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 2009.

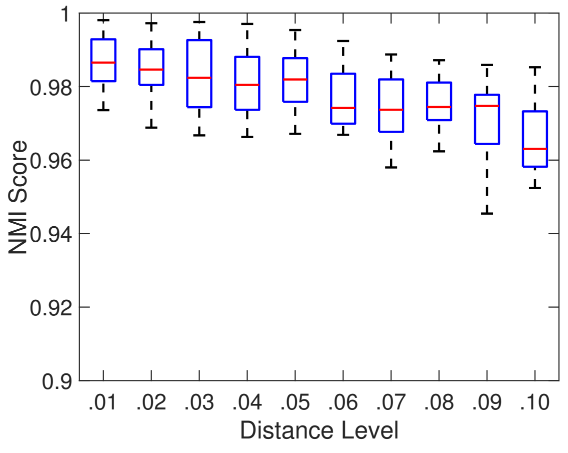
[4] R. C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010.

[5] L. Jiang, Y. Dong, N. Chen, and T. Chen. DACE: a scalable DP-means algorithm for clustering extremely large sequence data. *Bioinformatics*, 33(6):834–842, 2016.

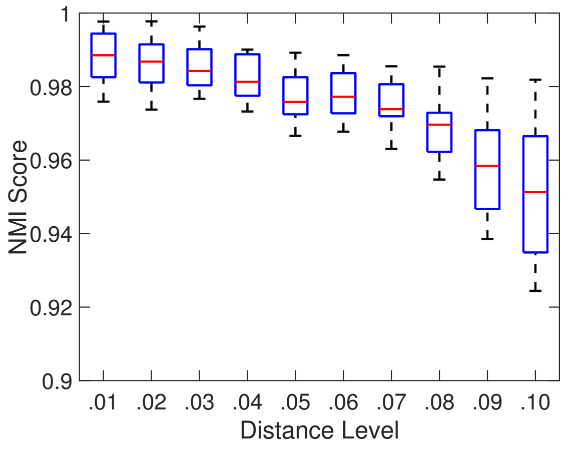
[6] F. Lin and W. W. Cohen. Power iteration clustering. In *Proceedings of the 27th International Conference on Machine Learning*, pages 655–662, 2010.

[7] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. MLlib: Machine learning in Apache Spark. *Journal of Machine Learning Research*, 17(34):1–7, 2016.

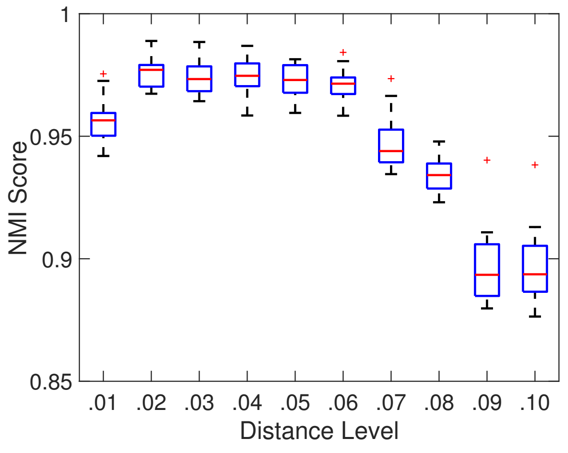
[8] J. F. M. Rodrigues and C. von Mering. HPC-CLUST: distributed hierarchical clustering for large sets of nucleotide sequences. *Bioinformatics*, 30(2):287–288, 2013.



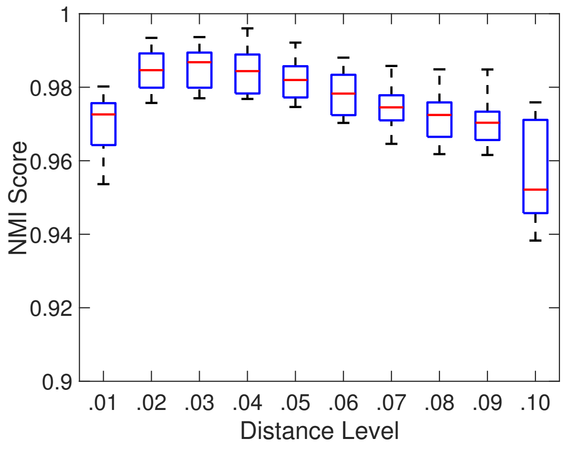
(a) UCLUST



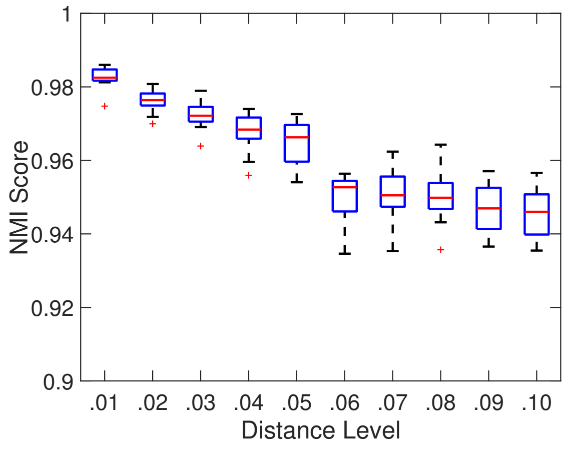
(b) Cd-hit



(c) Abundant OTU



(d) ESPRIT-Tree



(e) UCLUST applied to Greengenes

Figure 1: Averaged NMI scores obtained by comparing the clustering results generated by an OTU picking method with and without SLAD performed on (a-d) the plaque and (e) Greengenes datasets. For the experiments performed on the Greengenes dataset, only UCLUST finished in 72 hours, which is the wall-time limit of our computing cluster, so only UCLUST results are presented. At the 0.03 and 0.05 distance levels (the two commonly used thresholds for defining species- and genus-level OTUs, respectively, the NMI scores stay at a very high level ($0.97 \sim 0.99$) across all datasets and all tested methods.

[9] Y. Sun, Y. Cai, S. M. Huse, R. Knight, W. G. Farmerie, X. Wang, and V. Mai. A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis. *Briefings in Bioinformatics*, 13(1):107–121, 2011.

[10] Y. Sun, Y. Cai, L. Liu, F. Yu, M. L. Farrell, W. McKendree, and W. Farmerie. ESPRIT: estimating species richness using large collections of 16S rRNA pyrosequences. *Nucleic Acids Research*, 37(10):e76, 2009.