Canopy Clustering Algorithm for Large Metagenomic Data

Mohammad Arifur Rahman, Nathan LaPierre, Huzefa Rangwala and Daniel Barbara
Department of Computer Science
George Mason University
Fairfax, VA, United States
Email: mrahma23@gmu.edu, nathanl2012@gmail.com, rangwala@cs.gmu.edu and dbarbara@gmu.edu

***Abstract*—Recent sequencing technologies like Shotgun sequencing, Next generation sequencing etc. allow sequencing of genomes from samples taken directly from environments hosting multiple organisms that co-exist as communities within the ecological regions. This collective genomic process known as metagenomics has created the necessity of developing computational tools for the quantification of abundance, diversity, role of different species within communities, phenotype inferences etc. Enormous amount of data is created during metagenome sequencing process. Algorithms have been developed to cluster similar metagenome sequences. We have developed an approach for clustering metagenomic data that uses Canopy Clustering with Locality Sensitive Hashing distance approximation to make clustering process in metagenomic data faster. Canopy Clustering works as a preprocessing step to reduce pairwise distance calculation and enables efficient parallel processing with subsequent expensive cluster methods while LSH provides fast distance approximation and reduces data dimension. We tested our framework with three popular clustering mechanisms in literature on three synthetic and three real world large scale metagenome datasets and observed that our proposed approach can reduce runtime while providing similar in some cases better outcomes.**

***Keywords*-Clustering, Canopy, LSH, Metagenome**

I. Introduction

With earlier sequencing technology nearly 50 million bases of nucleotide sequence were available in public archives [1]. Now a single sequencing instrument can produce over 1 trillion nucleotide bases in just a single run [1]. Latest genome sequencing technologies like Shotgun and High Throughput sequencing have paved the way to Metagenomics - the study of genetic material recovered directly from samples that comprise organisms as co-existing communities. These samples can be taken from sea, soil and human body [2][3]. Metagenomics has enabled scientists to study all of the genomes in a community as a whole. Analysis of microbial community through metagenome data can reveal interesting relationships between microbial community and the host which in turn can lead to further investigation. For example analyzing metagenomic data from human gut microbiome can provide an understanding of the role played by microbes with regards to human health and disease [4].

[1]http://www.ncbi.nlm.nih.gov/genbank/statistics/

Instead of producing whole genomic sequences of the members of communities in host samples, latest sequencing technologies produce short contiguous subsequences called *reads* from random positions of actual whole genome. These reads from different organisms are commingled together posing fundamental challenge to further analysis of the data. Combining the reads of different organisms based on overlapping yet discriminating information from organism specific genome sequences is known as sequence assembly [5]. Many Sequence assemblers require reference genome sequence and the process of assembly is exceedingly complex, challenging and time consuming [6]. For this reason clustering metagenome data for identification of Operational Taxonomic Units (OTU) from 16s rRNA genes has become popular recently. 16S rRNA gene sequencing has been widely used for the analysis of genetic diversity within complex microbial communities. 16S sequences are marker genes, which exists in most microbial species but have variations in their sequences that allow them to be separated into different taxonomic groups [7]. OTU is used to classify groups of closely related individuals from similar or different taxonomic levels [8]. It is the most commonly used microbial diversity unit, especially when analyzing the small subunit 16S or 18S rRNA marker gene microbial datasets [9]. Sequences can be clustered according to their similarity to each other. Microbial OTUs are generally ecologically consistent across the hosts regardless of OTU clustering approaches [10].

Many Unsupervised and Supervised clustering approaches have been developed and used for the rapid analysis of large sets of whole and targeted 16S rRNA metagenomic sequences which are discussed in Section II - Literature Review. Analysis of microbiome datasets typically begins by clustering raw sequence reads and creating potential OTUs. Proper clustering of sequence reads assists in the metagenome assembly problem, allows computation of different species diversity metrics and allows further analysis with phenotype inferences. In this study we have proposed and evaluated a pre-clustering technique based on Canopy cluster method and Locality Sensitive Hashing. Our proposed framework can reduce pairwise comparison between sequences for similarity measure in large scale metagenome datasets by partitioning the dataset with fast LSH based

approximation. These initial partitions can be considered independent of each other. More accurate and expensive clustering methods can be deployed for each partition in parallel utilizing multi-core architechture of modern CPUs. Only the sequences inside a canopy will be considered for further sub-clustering not the sequences outside that canopy. This characteristic of Canopy clustering reduces expensive pairwise comparison significantly.

II. Literature Review

Analyzing metagenome sequences has poses great computational challenges since current DNA sequencing technologies generate hundreds of gigabytes of data in a single run and many computational methods have been developed over the years [11], [12]. Both availability of data and recent computational methods have enabled new detailed research into the human microbiome [13] and characterization of the Earth ecosystems microbiome, such as the Earth Microbiome Project (EMP)[2]. Metagenome dataset analysis usually begins with clustering raw biological sequence reads into operational taxonomic units (OTUs) based on sequence similarity. This process frequently referred to as OTU clustering also known as delineating. Focus of study is OTU clustering methods and ways to improve them in terms of OTU quality and runtime.

Some of the popular and early OTU clustering methods proposed were CD-HIT [14], DOTUR [15], MOTHUR [16] and ESPRIT [17]. Algorithm used in CD-HIT is a greedy incremental clustering algorithm and performs ordering on sequences. The ordering used in CD-HIT is determined by sequence length. CD-HIT sorts all sequences by length in decreasing order. The first sequence is considered as the representative of the first cluster. Rest of the sequences are compared with first sequence. If any of the sequences fall above the threshold for similarity, then it is grouped together with the first sequence. Subsequent comparisons have to fall above the threshold for similarity with all sequences in this cluster group. After this first round of cluster, the next ungrouped sequence is set as the representative of the next cluster. This process is repeated untill all sequences are part of a cluster. DOTUR, MOTHUR and ESPRIT exclusively use pairwise distance matrix as input and then perform hierarchical clustering clustering on 16S sequences. MOTHUR and DOTUR utilizes global alignment score between all pairs of sequences for pairwise distances computation. ESPRIT computes k-mer distance for each pair of input sequences, avoiding the expensive global alignment distance calculation. Number of sequence comparisons is also reduced by ESPRIT because of usage of heuristics. Hashing techniques have been utilized in earlier works for clustering metagenomic datasets. MC-LSH [18] utilizes an efficient locality sensitive based hashing function to approximate the pairwise sequence similarity. Similarity among sequences is computed based on randomly chosen indices that essentially compresses the input sequence. MC-MinH [19] algorithm uses the min-wise [20] hashing approach, along with a greedy clustering algorithm to group 16S and whole metagenomic sequences. Unequal length sequences are represented using contiguous subsequences or k-mers, and then pairwise similarity are approximated using independent min-wise hashing. Mash [21], uses MinHash locality sensitive hashing to reduce large sequences to a representative sketch and rapidly estimate pairwise distances between genomes or metagenomes. More recent methods for clustering whole metagenome sequence reads include TOSS [22], AbundanceBin [23], CompostBin [24], LikelyBin [25] and MetaCluster [26]. All unique k-mers are first clustered in TOSS then clusters are merged based on k-mer repetition.In AbundanceBin sequencing reads are modeled as mixture of Poisson distributions. Then Maximization (EM) algorithm is used to infer model parameters and final clustering. Principal component analysis is used in CompostBin to project the data into lower dimensional space and then uses the normalized cut clustering [27] algorithm to group sequences into taxa specific clusters. CompostBin also uses phylogenetic markers to assign clusters. LikelyBin uses an unsupervised, maximum-likelihood approach for clustering based on k-mer distributions of sequences. MetaCluster creates a top-down separation followed by a bottom-up merging for clustering metagenomic fragments based on k-mer frequencies and Spear-man distance computation. UCLUST [28] follows a greedy process and creates *seeds* of sequences which generate clusters based on percent identity. It is very fast but closed source software. Only the 32 version is made available for free with very limited documentations for academic and non-profit usage. 64-bit version of the software require expensive license. In last 2 years, 2 relatively new clustering methods have been introduced Swarm [29][30] and SUMACLUST [31]. These methods are some of the currently popular, relatively fast [32] and open source softwares. SWARM uses exhaustive single-linkage clustering based on optimal sequence alignment. Sequences that are less than a certain distance from any other other sequence in the cluster are clustered together. SUMACLUST follows a similar approach as UCLUST. Abundance-ordered list of input sequences are compared against the representative set of already-chosen sequences in SUMACLUST.

In this study we have introduced a very fast initial partitioning of large scale metagenomic datasets with Canopy Clustering [33] and Locality Sensitive Hashing [34][35][36]. This way efficient and scalable parallelism in OTU-clustering can be achieved that is capable of making existing OTU-clustering method multiple times faster by taking advantage of modern multicore CPU architectures. We have used UCLUST, SWARM and SUMACLUST with and without our proposed framework on 6 standard large

[2]http://www.earthmicrobiome.org/

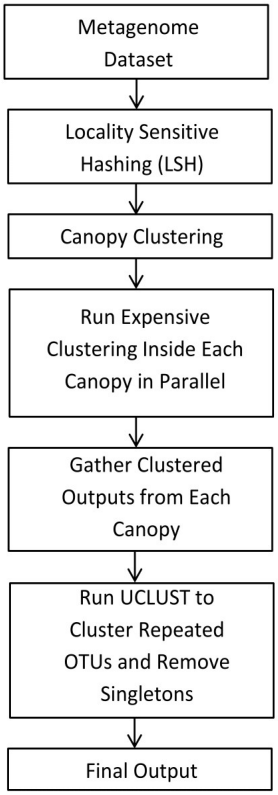


Figure 1. Overview of our proposed Canopy Clustering approach on Metagenome Data

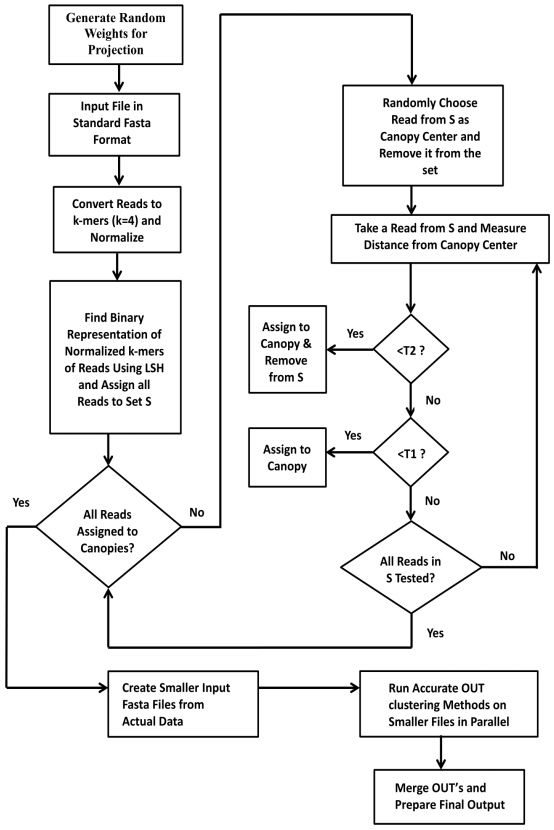


Figure 2. Workflow of LSH-Canopy for Large Scale Metagenome Data

scale metagenomic datasets to validate and compare our proposed approach. Our proposed framework has shown similar biodiversity metric values, higher F-measure values for datasets with known taxonomic profiles as ground truth and significantly less amount of runtime specially for large datasets.

III. Methods

A. Overview

In this section we provide an overview of our proposed framework. Figure 1 represents an abstract overview of our proposed approach. At first LSH codes will be created for each sequence reads and fast approximate distances will be calculated from these LSH codes to find Canopy memberships of each read. Then this membership information will be used to partition the large dataset for parallel sub-clustering using more expensive and accurate sequence clustering methods. Finally OTUs from these clusters will be merged by running a final clustering to eliminate repetitions due Canopies.

Figure 2 shows more detailed overview of our proposed LSH-Canopy clustering approach for metagenomic data. After computing the kmers and LSH code our proposed method will start canopy clustering based on the hamming distances between the binary representations of data points created by LSH. Once all reads in datasets are assigned to canopies, our method will create multiple partitions from original dataset based on canopy membership information. This will give one smaller input file per canopy for the subsequent expensive cluster methods. This way larger dataset will be partitioned and new smaller datasets will be created for those partitions. Finally each smaller input files representing a canopy will be provided as input to more accurate sequence clustering methods in parallel. In this study UCLUST, SUMACLUST and SWARM will be used as accurate clustering methods for canopies.

The final step of our proposed framework is to merge the clustered outputs from each canopy provided by more accurate and expensive clustering methods. There is a potential concern in this scenario. According to Canopy cluster algorithm a single data point may belong to multiple canopies as long as the soft threshold is met. As a result same sequence or collection of sequences in a metagenomic dataset can contribute may belong to multiple canopies thus contributing to multiple OTUs even though they are essentially same.

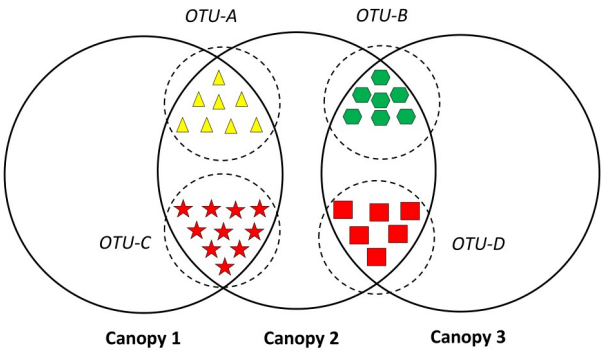


Figure 3. Same OTUs Occurring in Multiple Canopies

This scenario is depicted in Figure 3. The dashed circles titled OTU-A, B, C and D represent *potential OTUs*. Soild outer circles represent canopies titled Canopy-1, 2 and 3. OTU-A and C are shared by Canopy-1 and 2. OTU-B and D are shared by Canopy-2 and 3. The smaller triangles, stars, hexagons and rectangles represent sequence reads in metagenome data. Because of this repetitions another final clustering on resulting OTUs is required which will remove repetitions in OTU. We have chosen UCLUST for merging step since UCLUST is one of the fastest greedy sequence clustering methods. We chose 99% identity for merging UCLUST. Since we will be running this final clustering on already clustered OTU and subsequent taxonomy assignment is sensitive to minor changes in OTU, higher identity of UCLUST in this step will ensure that nearly same OTUs will be merged and variation is preserved.

In next sections we provide a brief description of Canopy Clustering, Locality Sensitive Hashing, their potential in metagenome data clustering, sub-clustering methods used in this study and our motivation for using them.

B. Canopy Clustering

Canopy Clustering was introduced by Andrew et al. [33]. It is often used as preprocessing step for other accurate and expensive clustering methods like K-means algorithm or the Hierarchical clustering algorithm. It is intended to speed up clustering operations on large data sets, where using another algorithm directly may be impractical in terms of run time and memory consumption for large number of pairwise distance calculations due to the size of the data set.

Canopy algorithm can greatly reduce the number of distance computations required for clustering by roughly partitioning the data into overlapping subsets. Then more accurate clustering can be performed inside the formed canopies by only measuring distances between pairs of data points that belong to same canopies. Points outside a canopy will not participate in clustering for that canopy. If a dataset contains total N instances then worst case calculations without canopy clustering is N^2. After canopy clustering if the number of canopies is k then worst case calculations with canopy clustering is $\sum_{i=1}^{k}(c_i)^2$ where c_i is the number of instances in ith canopy. The total difference g in pairwise distance calculations:

$$g = N^2 - \sum_{i=1}^{k}(c_i)^2 \tag{1}$$

Canopy clustering uses two basic distance thresholds namely *soft* threshold $T1$ and *tight* threshold $T2$. If data point p_1 is within the soft distance threshold $T1$ with another data point p_2 then p_1 will reside in same canopy as p_2 but p_1 may belong to other canopies too assuming that it has only met soft threshold and best match is yet to be found. Thus one data point may belong to multiple canopies in Canopy clustering. On the other hand if data point p_1 is within the tight distance threshold $T2$ with another data point p_2 then canopy clustering assigns p_1 to the same canopy as p_2 and stops assigning p_1 to any other canopy assuming that tight threshold has been met and best canopy assignment for p_1 has been found. In this case p_1 will not be repeated in other canopies. The process of canopy clustering begins with all the data points to be clustered. A point is randomly removed from the set, beginning a new canopy. All other data points are tested and assigned to the new canopy if the distance less than the soft threshold T_1. Moreover if the distance of the point is less than the tight threshold T_2 then it is removed from the original set. This process is repeated until there are no more data points in the set to cluster. These relatively cheaply clustered canopies can be clustered using a more expensive but accurate algorithm. A fast approximation which is discussed in next section will be used for Canopy membership identification. Once the dataset is partitioned other popular metagenome clustering can be used to sub-cluster canopies in parallel achieving both advantages of reduced calculations due to canopies and lower runtime due to proper utilization of parallel processing.

C. Locality Sensitive Hashing

A Locality Sensitive Hashing (LSH) [34][35][36] scheme is a distribution on a family $\mathcal{F}$ is defined for a metric space $\mathcal{M} = (M, d)$ a threshold $R > 0$ and an approximation factor $c > 1$. This family $\mathcal{F}$ is a family of functions $h : \mathcal{M} \to S$ which map elements from the metric space to a bucket $s \in S$. The LSH family satisfies the following conditions for any two points $p, q \in \mathcal{M}$, using a function $h \in \mathcal{F}$ which is chosen uniformly at random:

(i) If $d(p, q) \leq R$, then $h(p) = h(q)$ with probability at least P_1.

(ii) If $d(p, q) \geq R$, then $h(p) = h(q)$ with probability at most P_2.

A family is interesting when $P_1 > P_2$. Such a family $\mathcal{F}$ is called (R, cR, P_1, P_2)-sensitive. LSH has been used for fast comparison between points in very high dimensional

space [36]. In [37] data points were randomly projected to low dimensional bit signatures such that cosine distance is approximately preserved. LSH provides a convenient way of projecting high dimensional data points into low dimensional space and compute fast approximate distance between pairs of data points in this transformed low dimensional space. These characteristics of LSH make it appropriate to be used as a fast distance calculation mechanism between two reads in large scale metagenomic datasets.

LSH function based on random projection and bit sampling for hamming distance was introduced in [36]. This approach works for the Hamming distance over d-dimensional vectors $\{0,1\}^d$. Here, the family $\mathcal{F}$ of hash functions is the family of all the projections of points on one of the d coordinates, for example $\mathcal{F} = \{h : \{0,1\}^d \rightarrow \{0,1\} \mid h(x) = x_i \text{ for some } i \in \{1,...,d\}\}$, where x_i is the ith coordinate of x. A random function h from $\mathcal{F}$ selects a random bit from the input point. This family has the following parameters: $P_1 = 1 - R/d,\ P_2 = 1 - cR/d$. In this study we have chosen a similar random projection based hashing function that projects a d dimensional data point into a n dimensional bit representation where $n < d$. The intuition behind choosing $n < d$ is to reduce the actual data dimension for faster distance calculations. Afterwards we used Hamming distance between two projected data points with reduced dimensions to get approximate distance between them.

Given a random projection matrix v' of size $1 \times d$ and a vector a of size $1 \times d$ representing a single data point in dataset the dot product of these two vectors provides a scalar value. The sign of this value represents which side of the random hyperplane does the data point exist. This information can be represented with a bit. This way a single data with higher dimension can be represented with limited number of bits, one for each random hyperplane. Given these binary representations of two data points, the hamming distance between them says the number of hyperplanes that pass through the two data points. In other words this hamming distance says about the *disagreements* between any two data points which can be used for a rough and fast distance approximation. Considering matrix V of size $n \times d$ as a collection all random weights :

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,d} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,d} \end{bmatrix}$$

the dot products between each of the rows of matrix V and a data point a will result in a collection of n bits for that data point. In other words the d-dimensional data point will take the form of $\{0,1\}^n$. Hamming distance between 2 such binary number indicates the number of randomly generated hyperplanes for which 2 corresponding data points reside in opposing sides. This is the approximate distance measure that will be used for Canopy Clustering.

D. Sub-Clustering Inside Canopies

We have used three recent and popular sequence clustering methods as an expensive clustering measure inside canopies in this study. UCLUST [28], SUMACLUST [31] and SWARM [29] were used for sub-clustering canopies. UCLUST is closed-source software and provides very limited documentations[3]. The 32 bit version of UCLUST executable is available for academic usage. But the 64-bit versions which is necessary to handle large datasets, require expensive license. UCLUST follows a greedy process and creates *seeds* of sequences which generate clusters based on percent identity. SUMACLUST is an open-source software. It follows similar approach as UCLUST. Based on greedy strategy SUMACLUST incrementally constructs clusters by comparing an abundance-ordered list of input sequences against the representative set of already-chosen sequences. Initially this list is empty. Finally SWARM uses exhaustive single-linkage clustering based on optimal sequence alignment. Sequences that are less than a certain distance from any other other sequence in the cluster are clustered together. SWARM attempts to reduce the impact of clustering parameters on the resulting OTUs by avoiding arbitrary global clustering thresholds and input sequence ordering dependence. At first SWARM builds an initial set of OTUs is constructed by iteratively agglomerating similar amplicons. Then amplicon abundance values are used to reveal OTUs internal structures and to break them into sub-OTUs.

IV. Experimental Evaluations

A. Dataset Description

Different types of previously published mocked and real world genuine datasets were used for extensive evaluation of our approach and comparison with other methods. We have used three different previously published mock community data sets- three of them are 16S rRNA gene mock community data sets (Bokulich_2, Bokulich_3, and Bokulich_6) from [38]. We have also used three previously published natural data sets: a 16S rRNA gene soil data set (canadian_soil)from [39], a 16S rRNA gene human data set (body_sites) from [40], and an 18S rRNA gene soil data set (global_soil) from [41]. Statistics and relevant information regarding the datasets are provided in Table I. Description of the various dataset used in this paper is as follows:

Synthetic Datasets:

1) Bokulich_2: This dataset was prepared using the Illumina TruSeq v2 paired–end library preparation kit. It is a 16S rRNA gene mock microbial community data set. It has total 4 samples. For 2 of these samples abundance of

[3]http://www.drive5.com/uclust/uclust_userguide_1_1_579.pdf

Table I
DATASET STATISTICS

Datasets	Type	Reference	# of Reads	# of Samples	Read Length	Platform
Bokulich_2	Mock	[38]	6,938,836	4	189–251	HiSeq
Bokulich_3	Mock	[38]	3,594,237	4	114–151	HiSeq
Bokulich_6	Mock	[38]	250,903	1	114–150	HiSeq
Canadian Soil	Genuine	[39]	2,966,053	13	76–10	HiSeq
Body Sites	Genuine	[40]	886,630	602	117–351	GS FLX
Global Soil	Genuine	[41]	9,252,764	57	119–151	HiSeq

mock communities were distributed evenly. For the other 2 samples abundance of mock communities were inserted randomly. This dataset contains 19 taxonomic Families, 19 Genera, 22 Species and 22 Strains in total. This dataset can be found in the QIIME database under the study identifiers 1685.

2) Bokulich_3: Similar to Bokulich_2 except that it was sequenced in different environment and tool. The dataset was prepared with the TruSeq v1 paired-end library kit at Illumina Cambridge. Available at QIIME database under the study ID 1686.

3) Bokulich_6: A 16S rRNA dataset which was sequenced at Washington University School of Medicine. It has 1 sample which contains evenly distributed mock microbial community. This dataset contains 13 taxonomic Families, 23 Genera, 44 Species and 48 Strains in total.

All these datasets from Bokulich et al.[38] are available at QIIME database[4] under their respective ID's. Since these are mock datasets their taxonomic profile of microbiome are known.

Real World Datasets:

4) Canadian Soil: The Canadian Soil dataset[5] contains genomic data of Canadian soil spanning from Arctic Tundra to Agricultural soil suitable for different agricultural products like soi, wheat, corn etc. in different regions of Canada.

5) Body Sites: This dataset contains composition of bacterial communities from up to 27 body sites in healthy adults. A collection of 602 samples acquired from different body sites of human subjects are provided with meta-data.

6) Global Soil: Ramirez et al.[41] combined data representing a range of biomes from Alaska to Antarctica from two previous studies of Dinsdale EA et al.[42] and Lauber et al.[43] to compare with below-ground diversity in New York City's Central Park.

All of these datasets have been used in previous studies [38],[44]. Any analysis on these datasets requires appropriate preprocessing which can significantly affect results of clustering and taxonomic classification based on them [38]. Performance of different open source sequence clustering methods were assessed and compared in [44] on these datasets and they have also provided preprocessed and filtered using qiime and provided together (URL: https://github.com/ekopylova/otu-clustering).

[4]http://qiime.org/home_static/dataFiles.html
[5]http://www.cm2bl.org/

B. Evaluation Metrics

We have used some standard and popular performance measure to assess our proposed clustering method and compare with other clustering methods.

Faiths phylogenetic diversity metric (PD) - Faiths phylogenetic diversity [45] also known as PD is based on the Phylogeny tree. It combines all the branch lengths of the tree as a measure of diversity. So, if a new OTU is found and it is closely related to another OTU in the sample, it will be a small increase in diversity. However, if a new OTU is found and it comes from a totally different lineage than anything else in the sample, it will contribute a lot to increasing the diversity:

Shannon Entropy - Shannon-Wiener diversity index is defined as:

$$H = -\sum_{i=1}^{s} (p_i \log_2 p_i) \tag{2}$$

where s is the number of OTU and p_i is the proportion of the community represented by OTU i.

Simpson's Index - Simpsons index is defined as $1 - dominance$ or

$$1 - \sum p_i^2 \tag{3}$$

where where p_i is the proportion of the community represented by OTU i.

F-measure - For synthetic datasets, false-positive (FP; taxonomy/OTU string exists in observed but not expected), false-negative (FN; taxonomy/OTU string exists in expected but not observed), and true-positive (TP; taxonomy/OTU string exists in both observed and expected) measures were computed from cluster output and the ground truth or expected taxonomic composition. The following definitions were used:

$$precision = \frac{TP}{(TP + FP)} \tag{4}$$

$$recall = \frac{TP}{(TP + FN)} \tag{5}$$

$$FScore = \frac{2 \times precision \times recall}{(precision + recall)} \tag{6}$$

C. Experimental Details

For kmers, the value of parameter k was set to 4. Which means occurrences of all possible combination of 4 letter nucleotides were generated from the reads of datasets. Since $k = 4$ and there can be 4 possible nucleotide representatives (A,C,G and T), total number of features became $4^4 = 256$. This way we were able to convert the string representation of nucleotide data in standard FASTA format into numeric. After finding the kmers, each numeric representation of data reads were normalized to range $[0 - 1]$.

For Locality Sensitive Hashing the number of Hyperplanes (parameter d) was set to 150. One of the motivations of using LSH was fast approximation of distances from one data read to another while reducing feature dimension. Since total number of features became 256 after numeric representation using $kmer$, we wanted to reduce the number of features while maintaining a relatively accurate yet fast distance approximation from LSH. So the value of parameter d should be less than the actual number of feature but not too small because that might lead to wrong distance approximation.

The two parameters of Canopy clustering namely $T1$ (loose threshold) and $T2$ (tight threshold) based on the reduced feature space provided by LSH and normalized hamming distance measure define which data reads will be grouped initially before the preferred expensive clustering begins. For this experiment and comparison we have chosen $T1 = 0.46$ and $T2 = 0.34$. For parameter $d = 150$, canopy threshold $T1 = 0.46$ refers that a data read can belong to a canopy if the hamming distance is $150 \times 0.46 = 69$ or less. And threshold $T2 = 0.34$ refers that if the hamming distance between randomly chosen canopy centroid and data read is $150 \times 0.34 = 51$ or less then the read should not be included in any other canopy assuming that we have found a stable canopy membership for the read. Once canopy memberships of all reads were found we created multiple smaller data files in standard FASTA format from actual dataset to be used by expensive cluster.

We performed all the experiments on computers with Intel 5th generation Core i7 2.70GHz 64bit processor with 8 core CPUs and 12GB memory. For implementation we used Python 2.7.12 and QIIME [46] version 1.9.0 - a popular open source Bioinformatics pipeline that combines many metagenome clustering methods including the ones which has been used in this work. One important aspect our proposed approach is parallelism. Each canopy can be clustered in parallel. To take full advantage of today's multi-core CPU based computing systems we utilized Python's *multiprocessing* module instead of *threading* module. According to Python's documentation[6] *multiprocessing* module allows the programmer to fully leverage multiple processors on a given machine by spawning subprocesses instead of threads. In our experiment these subprocesses are independent Python programs running more accurate and expensive clustering on smaller datasets created by canopy clusterings. In our experiment the number of parallel processes was set equal to the number of CPU cores (which is 8 in our case) with an expectation that OS will deploy each of those processes in separate cores unlike threading. This is convenient since it utilizes 100% of the CPU in most cases.

After receiving output from all canopy clustering we merged the outputs to make a single clustering result. Some performance metric used in this study like Faiths Phylogenetic Diversity (PD) metric require sequence alignment. We used PyNast[7] [47] open source sequence aligner for aligning clustered output.

V. Results Discussion

A. *Clustering Performance Comparison*

Performance of UCLUST [28], SUMACLUST [31], SWARM [29] and their respective versions with our proposed approach Locality Sensitive Hashing based Canopy Clustering (LSH-Canopy) is provided in Table II. Datasets used in this study contain multiple samples. Performance metrics like Phylogenetic Diversity (PD), Shannon entropy and Simpson index are generated per sample basis for any dataset. To show comparative performance of the clustering methods used in this study we showed the range of values the performance metrics in $[min - max]$ where min and max represent the minimum and maximum values respectively observed in the whole dataset regardless of samples for a particular clustering method. This represents the range of values a performance metric can take over all samples in a dataset. Performance of 2 clustering methods on same dataset can be assumed similar in terms of biodiversity if their $minimum$ and $maximum$ values for performance metrics are similar. Metrics representing biodiversity can significantly change over clustering methods, their respective data pre-processing pipeline, post processing, range of OTU lengths, number of OTU's etc. Nevertheless, $[min - max]$ range of a clustering method and it's counterpart with our proposed LSH-Canopy pipeline should be roughly similar. F-measure and Pearson Correlation Coefficient (ρ-value) can take only one value each for a clustering method on a dataset. And the correlation metric (ρ-value) is applicable for clustering methods with our proposed approach only since this is a correlation between the output of a standard clustering method and it's respective LSH-Canopy version. Thus we can interpret a better outcome of this experiment as (i) similar range of biodiversity (PD, Shannon and Simpson values), (ii) higher F-measure and Correlation (ρ-value) with respect to corresponding clustering without our proposed LSH-Canopy pipeline and finally (iii) lower running time.

[6]https://docs.python.org/2/library/multiprocessing.html

[7]http://biocore.github.io/pynast/

From Table II we can see that UCLUST and LSH-Canopy-UCLUST provide similar range for PD, Shannon and Simpson values for both synthetic and real world datasets. Moreover, LSH-Canopy-UCLUST provided higher F-score than UCLUST for all 3 synthetic datasets used in this study. The lowest F-score observed by UCLUST was 0.39 for Bokulich_2 dataset. with LSH-Canopy-UCLUST we achieved 0.42 for the same dataset. For synthetic datasets taxonomic profiles were known as prior. This taxonomic distribution can be taken as ground truth and it can be used to identify whether the cluster representatives actually support this ground truth. A higher F-score is an indicator that relevant and important cluster representatives are retained as they support ground truth. Higher F-scores of our proposed approach is due to the nature of how canopy is formed. A single data instance can belong to multiple canopies due to the nature of soft threshold $T1$. As a result a single read can contribute to multiple Operational Taxonomic Units (OTU) in metagenome data. As a result even after removing singletons, expensive clustering inside a canopy can bring higher numbers of useful OTU's. For real world datasets LSH-Canopy-UCLUST provided similar range of values for PD, Shannon and Simpson. Correlation of LSH-Canopy-UCLUST with navie UCLUST at the Genus level of assigned taxonomy was observed to be highly correlated. The lowest ρ-value was 0.8177 for Canadian Soil metagenomic data which is the largest dataset used in this study containing more than 9 million reads.

Our proposed approach also improves the outcomes of SUMACLUST. From Table II we can see that SUMACLUST provided higher F-score than UCLUST for Bokulich_2 and Bokulich_3 and same value for Bokulich_6. After running SUMACLUST with LSH-Canopy we observed same F-score as SUMACLUST for Bokulich_2 and higher for other 2 synthetic datasets. SUMACLUST with our proposed approach outperforms the naive SUMACLUST and UCLUST. Ranges of PD, Shannon and Simpson were similar to naive SUMACLUST for both synthetic and real world datasets. ρ-values were higher indicating a strong positive correlation with naive SUMACLUST specially for synthetic datasets. Lowest ρ-value was 0.7859 which was observed for larger Canadian Soil real world metagenome dataset.

Finally SWARM with our proposed LSH-Canopy pipeline also showed better results than naive SWARM. We observed the highest F-score of 0.56 in this study for Bokulich_6 dataset from both naive and LSH-Canopy version of SWARM. For the other 2 synthetic datasets our proposed LSH-Canopy version of SWARM outperforms the naive counterpart. Ranges for PD, Shannon and Simpson were similar and correlation values were observed to be high. The lowest ρ-value for LSH-Canopy-SWARM was 0.7736 for Canadian Soil dataset.

B. Runtime Comparison

Table III shows the runtime in minutes of UCLUST, SUMACLUST, SWARM and their respective version with our proposed LSH-Canopy pipeline. Time information in Table III was recorded while populating performance metric values for Table II for the parameter setting mentioned in Experimental Details section. Improving runtime was one of the major motivations of our proposed approach - fast distance approximation using LSH, using this approximation for initial canopy clustering to reduce pairwise comparison and finally utilize parallelism in multi-core CPU. We can see from Table III that LSH-Canopy-UCLUST outperforms UCLUST for all datasets. In fact LSH-Canopy-UCLUST took nearly half the time or less of naive UCLUST for all datasets. This improvement in runtime is due to efficient parallelization of expensive clustering in canopies. For Canadian Soil dataset which is the largest dataset used in this study and contains more than 9 million reads, UCLUST takes 72.47 minutes whereas LSH-Canopy-UCLUST takes only 41.92 minutes.

For SUMACLUST we can see from Table III that LSH-Canopy-SUMACLUST outperforms naive SUMACLUST in most cases. Major improvement in runtime was observed in large scale real world datasets specially Canadian Soil and Global Soil datasets. The highest amount of time taken by naive SUMACLUST is 510.92 minutes where LSH-Canopy-SUMACLUST performs a similar job in just 64.58 minutes. Which is almost 8 times faster than the naive version. But LSH-Canopy-SUMACLUST took more time than SUMACLUST for Bokulich_6 dataset which is the smallest dataset used in this study with only 250,903 number of reads. This gives the intuition that for relatively smaller datasets applying Locality Sensitive Hashing followed Canopy clustering might be unnecessary and may incur more time than the naive version of the clustering methods. A greedy clustering is enough and faster than a modified or added pipeline.

We can see a similar footprints for LSH-Canopy-SWARM from Table III. SWARM took 128.12 minutes for Bokulich_2 dataset whereas LSH-Canopy-SWARM took 37.87 minutes only. SWARM was relatively faster for Bokulich_3, Bokulich_6 and Body Site datasets than SUMACLUST and LSH-Canopy-SWARM was faster than the corresponding naive version. But again major improvement in runtime was observed for larger real world datasets namely Canadian Soil and Global Soil datasets. LSH-Canopy-SWARM was nearly 3 times faster than naive SWARM for Canadian Soil dataset and **??** faster than Global Soil dataset. These observations refers to the fact that our proposed LSH-Canopy pipeline scale well with larger datasets and performs better in such case comparing to medium or smaller datasets.

Table II
PERFORMANCE COMPARISON

Methods	Comparison Metric	Datasets					
		Synthetic			*Real World*		
		Bokulich_2	Bokulich_3	Bokulich_6	Body Sites	Canadian Soil	Global Soil
UCLUST	F-Measure	0.39	0.40	0.51	N/A	N/A	N/A
	PD	$[171.95 - 221.85]$	$[186.90 - 212.84]$	$[2.98 - 3.29]$	$[1.46 - 46.79]$	$[0.30 - 1352.73]$	$[0.30 - 1352.73]$
	Shannon	$[2.52 - 3.51]$	$[2.43 - 3.54]$	$[5.87 - 5.87]$	$[0.29 - 7.67]$	$[2.32 - 10.85]$	$[1.84 - 8.30]$
	Simpson	$[0.55 - 0.75]$	$[0.55 - 0.76]$	$[0.96 - 0.96]$	$[0.049 - 0.98]$	$[0.8 - 0.99]$	$[0.0 - 0.98]$
LSH-Canopy-UCLUST	F-Measure	***0.42***	***0.44***	***0.54***	N/A	N/A	N/A
	ρ–value	***0.9911***	***0.9935***	***0.9787***	***0.9581***	***0.8177***	***0.9824***
	PD	$[162.49 - 206.72]$	$[168.01 - 194.66]$	$[109.33 - 109.33]$	$[2.37 - 44.52]$	$[0.52 - 1319.31]$	$[3.02 - 3.26]$
	Shannon	$[2.26 - 3.94]$	$[2.71 - 4.86]$	$[6.2 - 6.2]$	$[0.75 - 7.06]$	$[3.16 - 11.94]$	$[4.36 - 11.48]$
	Simpson	$[0.64 - 0.77]$	$[0.58 - 0.83]$	$[0.96 - 0.96]$	$[0.097 - 0.99]$	$[0.88 - 0.99]$	$[0.11 - 0.99]$
SUMACLUST	F-Measure	0.40	0.41	0.51	N/A	N/A	N/A
	PD	$[106.00 - 162.78]$	$[142.85 - 174.19]$	$[89.22 - 89.22]$	$[0.93 - 39.47]$	$[0.59 - 1279.29]$	$[2.98 - 3.29]$
	Shannon	$[2.00 - 3.01]$	$[2.19 - 3.28]$	$[5.48 - 5.48]$	$[0.16 - 7.43]$	$[2.32 - 10.32]$	$[1.00 - 8.03]$
	Simpson	$[0.52 - 0.73]$	$[0.54 - 0.75]$	$[0.95 - 0.95]$	$[0.027 - 0.98]$	$[0.8 - 0.99]$	$[0.40 - 0.98]$
LSH-Canopy-SUMACLUST	F-Measure	***0.40***	***0.43***	***0.53***	N/A	N/A	N/A
	ρ–value	***0.9909***	***0.9950***	***0.9726***	***0.9575***	***0.7859***	***0.8812***
	PD	$[112.36 - 177.37]$	$[157.55 - 183.41]$	$[91.96 - 91.96]$	$[0.74 - 39.51]$	$[0.86 - 1281.66]$	$[1.05 - 5.14]$
	Shannon	$[3.15 - 4.02]$	$[2.6 - 3.65]$	$[5.77 - 5.77]$	$[0.18 - 7.92]$	$[2.52 - 11.62]$	$[2.73 - 9.91]$
	Simpson	$[0.61 - 0.75]$	$[0.57 - 0.79]$	$[0.97 - 0.97]$	$[0.05 - 0.99]$	$[0.90 - 0.99]$	$[0.31 - 0.98]$
SWARM	F-Measure	0.47	0.48	0.56	N/A	N/A	N/A
	PD	$[18.37 - 24.73]$	$[17.36 - 19.81]$	$[30.84 - 30.84]$	$[1.44 - 28.66]$	$[0.54 - 706.57]$	$[5.79 - 6.18]$
	Shannon	$[2.98 - 3.91]$	$[2.01 - 3.04]$	$[5.03 - 5.03]$	$[0.28 - 7.63]$	$[1.0 - 9.69]$	$[1.66 - 8.30]$
	Simpson	$[0.70 - 0.82]$	$[0.53 - 0.74]$	$[0.95 - 0.95]$	$[0.05 - 0.98]$	$[0.5 - 0.99]$	$[0.00 - 0.99]$
LSH-Canopy-SWARM	F-Measure	***0.54***	***0.55***	***0.56***	N/A	N/A	N/A
	ρ–value	***0.9997***	***0.9980***	***0.9463***	***0.9769***	***0.7736***	***0.9136***
	PD	$[17.53 - 25.79]$	$[17.48 - 25.92]$	$[32.17 - 32.17]$	$[1.34 - 27.65]$	$[0.10 - 745.40]$	$[2.17 - 7.41]$
	Shannon	$[2.66 - 4.12]$	$[2.19 - 3.62]$	$[5.15 - 5.15]$	$[0.63 - 7.39]$	$[1.12 - 11.31]$	$[3.14 - 12.17]$
	Simpson	$[0.76 - 0.86]$	$[0.49 - 0.78]$	$[0.97 - 0.97]$	$[0.08 - 0.99]$	$[0.64 - 0.99]$	$[0.38 - 0.99]$

Table shows values of F-measure for mock datasets for which taxonomy is known, Phylogeny Diversity (PD) value, Shannon and Simpson Coefficient intervals in the format $[min - max]$. Most of these datasets contain multiple samples and PD, Shannon and Simpson values are generated for each of these samples separately. So the values are represented as intervals from minimum to maximum that covers all performance metric values for the whole dataset regardless of number samples.

Table III
RUNTIME COMPARISON (IN MINUTES)

Datasets			Methods								
Type	Title	# of Reads	UCLUST	UCLUST with LSH-Canopy	Speed Up	SUMACLUST	SUMACLUST with LSH-Canopy	Speed Up	SWARM	SWARM with LSH-Canopy	Speed Up
Synthetic	Bokulich_2	6,938,836	10.08	***4.82***	***2.09x***	87.53	***33.03***	***2.65x***	128.12	***37.87***	***3.38x***
	Bokulich_3	3,594,237	7.85	***3.65***	***2.15x***	11.73	***7.89***	***1.48x***	9.10	***6.93***	***1.31x***
	Bokulich_6	250,903	1.73	***0.95***	***1.82x***	***1.28***	1.34	0.96	1.97	***1.29***	***1.53x***
Real World	Body Sites	886,630	2.06	***0.98***	***2.10x***	15.42	***7.15***	***2.16x***	3.64	***1.17***	***3.11x***
	Canadian Soil	2,966,053	9.55	***5.32***	***1.80x***	363.96	***79.45***	***4.58x***	97.5	***31.34***	***3.11x***
	Global Soil	9,252,764	72.47	***41.92***	***1.73x***	510.92	***64.58***	***7.91x***	269.07	***53.71***	***5.00x***

C. *Effect of Varying Number of Multi-processes*

Figure 4a-4f shows how the number of multiprocess can affect runtime of the LSH-Canopy version of UCLUST, SUMACLUST and SWARM on 6 datasets that we have used in this study. We can see that increasing the number of multiprocess to perform expensive clustering inside canopies in parallel can significantly reduce total runtime. The steepest curve showing reduction in runtime can be found in Figure 4e for LSH-Canopy-SUMACLUST on Canadian Soil dataset. For single process version LSH-Canopy-SUMACLUST is slower in most cases except Bokulich_2 dataset where it's runtime is better comparing to SWARM with LSH-Canopy. For Body Site dataset (Figure 4d) the LSH-Canopy version of SWARM and UCLUST have closer runtime with respect to increasing number of multiprocess. Figure 4f shows reduction in runtime for the largest dataset used in this study namely Global Soil with more than 9 million reads. Even though there are significant difference in the runtime of single process version, LSH-Canopy reduces runtime to closer interval for UCLUST, SWARM and SUMACLUST with increasing number of multiprocess.

VI. CONCLUSION AND FUTURE WORK

We propose a framework that can be used as pre-clustering for any accurate and relatively expensive clustering on large scale metagenomic datasets. Our proposed approach scales well with large datasets and provide significant reduction in computation time. We demonstrate that our approach provides similar outcome in terms of biodiversity metrics used in this study, scores based on ground truth and taxonomic correlation with corresponding relatively expensive cluster. Our approach takes advantage of the multi-core CPU systems by partitioning the large dataset roughly with fast and cheaper pairwise distance measure and then deploying comparatively expensive clustering in

parallel which considers only data points that are inside the partition. We plan to develop standalone cluster mechanism for the canopies in future.

REFERENCES

[1] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?" *PLoS Biol*, vol. 13, no. 7, p. e1002195, 2015.

[2] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson *et al.*, "Environmental genome shotgun sequencing of the sargasso sea," *science*, vol. 304, no. 5667, pp. 66–74, 2004.

[3] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada *et al.*, "A human gut microbial gene catalogue established by metagenomic sequencing," *nature*, vol. 464, no. 7285, pp. 59–65, 2010.

[4] P. J. Turnbaugh, R. E. Ley, M. Hamady, C. Fraser-Liggett, R. Knight, and J. I. Gordon, "The human microbiome project: exploring the microbial part of ourselves in a changing world," *Nature*, vol. 449, no. 7164, p. 804, 2007.

[5] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington *et al.*, "A whole-genome assembly of drosophila," *Science*, vol. 287, no. 5461, pp. 2196–2204, 2000.

[6] A. Charuvaka and H. Rangwala, "Evaluation of short read metagenomic assembly," *BMC genomics*, vol. 12, no. 2, p. 1, 2011.

[7] S. Chakravorty, D. Helb, M. Burday, N. Connell, and D. Alland, "A detailed analysis of 16s ribosomal rna gene segments for the diagnosis of pathogenic bacteria," *Journal of microbiological methods*, vol. 69, no. 2, pp. 330–339, 2007.

[8] M. Blaxter, J. Mann, T. Chapman, F. Thomas, C. Whitton, R. Floyd, and E. Abebe, "Defining operational taxonomic units using dna barcode data," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 360, no. 1462, pp. 1935–1943, 2005.

[9] A. F. Koeppel and M. Wu, "Surprisingly extensive mixed phylogenetic and ecological signals among bacterial operational taxonomic units," *Nucleic acids research*, p. gkt241, 2013.

[10] T. S. Schmidt, J. F. M. Rodrigues, and C. Von Mering, "Ecological consistency of ssu rrna-based operational taxonomic units at a global scale," *PLoS Comput Biol*, vol. 10, no. 4, p. e1003594, 2014.

[11] Hugenholtz, Philip and Tyson, Gene W., "Microbiology: Metagenomics," *Nature*, vol. 455, no. 0028-0836, pp. 481–483, 2008.

[12] Pop, Mihai and Salzberg, Steven L., "Bioinformatics challenges of new sequencing technology," *Trends in Genetics*, vol. 24, no. 0168-9525, pp. 142–149, 2008.

[13] The Human Microbiome Project Consortium, "A framework for human microbiome research," *Nature*, vol. 486, no. 0028-0836, pp. 215–221, 2012.

[14] Li, Weizhong and Godzik, Adam, "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.

[15] Schloss, Patrick D. and Handelsman, Jo, "Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness," *Applied and Environmental Microbiology*, vol. 71, no. 3, pp. 1501–1506, 2005.

[16] Schloss, Patrick D. and Westcott, Sarah L. and Ryabin, Thomas and Hall, Justine R. and Hartmann, Martin and Hollister, Emily B. and Lesniewski, Ryan A. and Oakley, Brian B. and Parks, Donovan H. and Robinson, Courtney J. and Sahl, Jason W. and Stres, Blaz and Thallinger, Gerhard G. and Van Horn, David J. and Weber, Carolyn F., "Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities," *Applied and Environmental Microbiology*, vol. 75, no. 23, pp. 7537–7541, 2009.

[17] Sun, Yijun and Cai, Yunpeng and Liu, Li and Yu, Fahong and Farrell, Michael L. and McKendree, William and Farmerie, William, "Esprit: estimating species richness using large collections of 16s rrna pyrosequences," *Nucleic Acids Research*, 2009.

[18] Z. Rasheed, H. Rangwala, and D. Barbara, "Efficient clustering of metagenomic sequences using locality sensitive hashing." in *SDM*. SIAM, 2012, pp. 1023–1034.

[19] Z. Rasheed and H. Rangwala, *MC-MinH: Metagenome Clustering using Minwise based Hashing*, ch. 74, pp. 677–685. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611972832.75

[20] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, no. 3, pp. 630–659, 2000.

[21] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy, "Mash: fast genome and metagenome distance estimation using minhash," *bioRxiv*, p. 029827, 2016.

[22] O. Tanaseichuk, J. Borneman, and T. Jiang, "Separating metagenomic short reads into genomes via clustering," *Algorithms for Molecular Biology*, vol. 7, no. 1, p. 1, 2012.

[23] Y.-W. Wu and Y. Ye, "A novel abundance-based algorithm for binning metagenomic sequences using l-tuples," *Journal of Computational Biology*, vol. 18, no. 3, pp. 523–534, 2011.

[24] S. Chatterji, I. Yamazaki, Z. Bai, and J. A. Eisen, "Compostbin: A dna composition-based algorithm for binning environmental shotgun reads," in *Annual International Conference on Research in Computational Molecular Biology*. Springer, 2008, pp. 17–28.

[25] A. Kislyuk, S. Bhatnagar, J. Dushoff, and J. S. Weitz, "Unsupervised statistical clustering of environmental shotgun sequences," *BMC bioinformatics*, vol. 10, no. 1, p. 1, 2009.

[26] B. Yang, Y. Peng, H. Leung, S.-M. Yiu, J. Qin, R. Li, and F. Y. Chin, "Metacluster: unsupervised binning of environmental genomic fragments and taxonomic annotation," in *Proceedings of the first ACM international conference on bioinformatics and computational biology*. ACM, 2010, pp. 170–179.

[27] M. Van Den Heuvel, R. Mandl, and H. H. Pol, "Normalized cut group clustering of resting-state fmri data," *PloS one*, vol. 3, no. 4, p. e2001, 2008.

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

[29] F, Mah and T, Rognes and C, Quince and C, de Vargas and M, Dunthorn, "Swarm: robust and fast clustering method for amplicon-based studies," *PeerJ*, vol. 2:e593, 2014.

[30] F. Mahé, T. Rognes, C. Quince, C. De Vargas, and M. Dunthorn, "Swarm v2: highly-scalable and high-resolution amplicon clustering," *PeerJ, volume=3, pages=e1420, year=2015, publisher=PeerJ Inc.*

[31] C, Mercier and F, Boyer and A, Bonin and E, Coissac, "Sumatra and sumaclust: fast and exact comparison and clustering of sequences [submitted for publication]," *Available at https://git.metabarcoding.org/obitools/sumatra/wikis/home*, 2014.

[32] E. Kopylova, J. A. Navas-Molina, C. Mercier, Z. Z. Xu, F. Mahé, Y. He, H.-W. Zhou, T. Rognes, J. G. Caporaso, and R. Knight, "Open-source sequence clustering methods improve the state of the art," *mSystems*, vol. 1, no. 1, pp. e00 003–15, 2016.

[33] McCallum, Andrew and Nigam, Kamal and Ungar, Lyle H., "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: ACM, 2000, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/347090.347123

[34] Rajaraman, A. and Ullman, J, *Mining of Massive Datasets, Ch. 3*, 2010.

[35] Gionis, Aristides and Indyk, Piotr and Motwani, Rajeev, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 518–529. [Online]. Available: http://dl.acm.org/citation.cfm?id=645925.671516

[36] Indyk, Piotr and Motwani, Rajeev, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: ACM, 1998, pp. 604–613. [Online]. Available: http://doi.acm.org/10.1145/276698.276876

[37] Charikar, Moses S., "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: ACM, 2002, pp. 380–388. [Online]. Available: http://doi.acm.org/10.1145/509907.509965

[38] Bokulich, NA and Subramanian, S and Faith JJ and Gevers, D and Gordon, JI and Knight, R and Mills, DA and Caporaso, JG, "Quality-filtering vastly improves diversity estimates from illumina amplicon sequencing," *Nat Methods*, vol. 10, pp. 57–59, 2013.

[39] JD, Neufeld and K, Engel and J, Cheng and G, Moreno-Hagelsieb and DR, Rose and TC,Charles, "Open resource metagenomics: a model for sharing metagenomic libraries," *Standards in Genomic Sciences*, vol. 5, pp. 203–210, 2011.

[40] Costello, Elizabeth K. and Lauber, Christian L. and Hamady, Micah and Fierer, Noah and Gordon, Jeffrey I. and Knight, Rob, "Bacterial community variation in human body habitats across space and time," *Science*, vol. 326, no. 5960, pp. 1694–1697, 2009.

[41] Ramirez, Kelly S. and Leff, Jonathan W. and Barberán, Albert and Bates, Scott Thomas and Betley, Jason and Crowther, Thomas W. and Kelly, Eugene F. and Oldfield, Emily E. and Shaw, E. Ashley and Steenbock, Christopher and Bradford, Mark A. and Wall, Diana H. and Fierer, Noah, "Biogeographic patterns in below-ground diversity in new york city's central park are similar to those observed globally," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 281, no. 1795, 2014.

[42] et al. Dinsdale EA, "Functional metagenomic profiling of nine biomes," *Nature*, vol. 452, pp. 629–632, 2008.

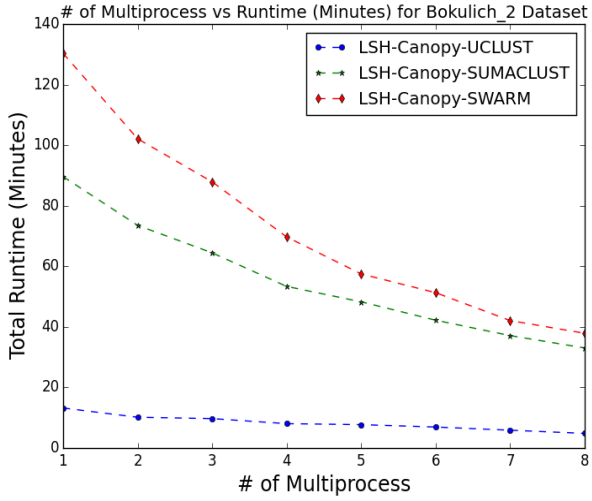
[43] Lauber, Christian L. and Hamady, Micah and Knight, Rob and Fierer, Noah, "Pyrosequencing-based assessment of soil ph as a predictor of soil bacterial community structure at the continental scale," *Applied and Environmental Microbiology*, vol. 75, no. 15, pp. 5111–5120, 2009.

[44] Kopylova, Evguenia and Navas-Molina, Jose A. and Mercier, Céline and Xu, Zhenjiang Zech and Mahé, Frédéric and He, Yan and Zhou, Hong-Wei and Rognes, Torbjørn and Caporaso, J. Gregory and Knight, Rob, "Open-source sequence clustering methods improve the state of the art," *mSystems*, vol. 1, no. 1, 2016.

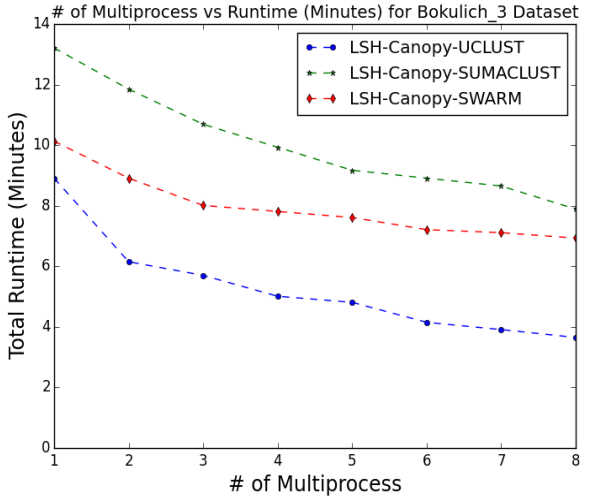
[45] Faith, Daniel P, "Conservation evaluation and phylogenetic diversity," *Biological conservation*, vol. 61, no. 1, pp. 1–10, 1992.

[46] Caporaso, J Gregory and Kuczynski, Justin and Stombaugh, Jesse and Bittinger, Kyle and Bushman, Frederic D and Costello, Elizabeth K and Fierer, Noah and Pena, Antonio Gonzalez and Goodrich, Julia K and Gordon, Jeffrey I and Huttley, Gavin A and Kelley, Scott T and Knights, Dan and Koenig, Jeremy E and Ley, Ruth E and Lozupone, Catherine A and McDonald, Daniel and Muegge, Brian D and Pirrung, Meg and Reeder, Jens and Sevinsky, Joel R and , Peter J and Walters, William A and Widmann, Jeremy and Yatsunenko, Tanya and Zaneveld, Jesse and Knight, Rob, "Qiime allows analysis of high-throughput community sequencing data," *Nat Meth*, vol. 7, no. 5, pp. 335–336, 2010.

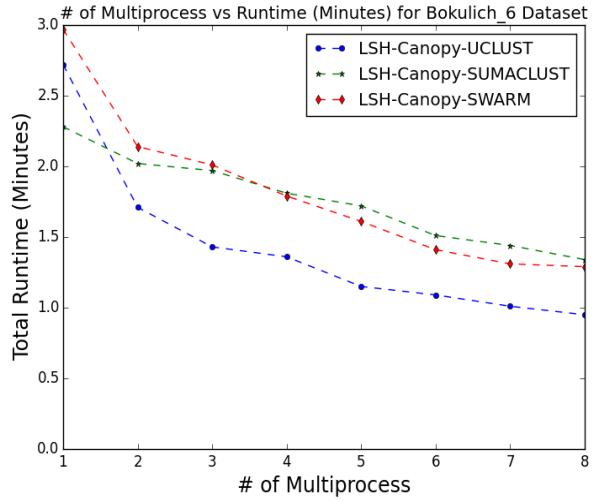
[47] Caporaso, J. Gregory and Bittinger, Kyle and Bushman, Frederic D. and DeSantis, Todd Z. and Andersen, Gary L. and Knight, Rob, "Pynast: a flexible tool for aligning sequences to a template alignment," *Bioinformatics*, vol. 26, no. 2, pp. 266–267, 2010.



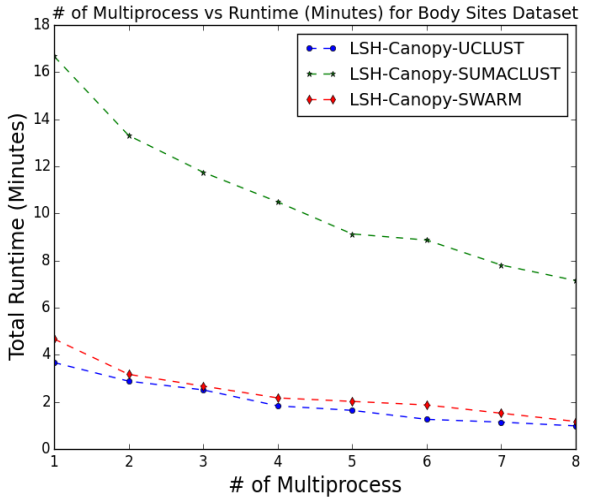
(a)



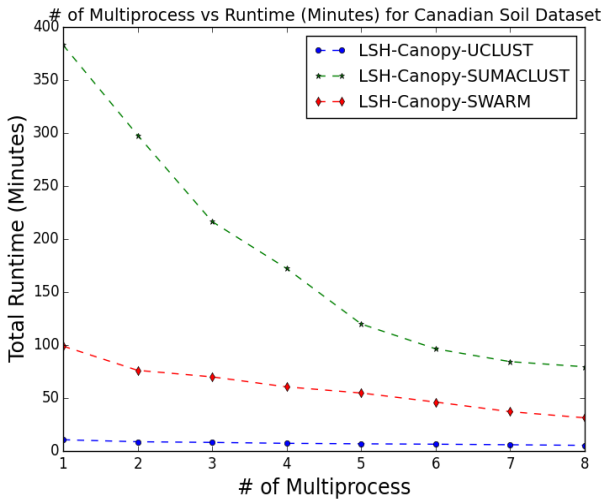
(b)



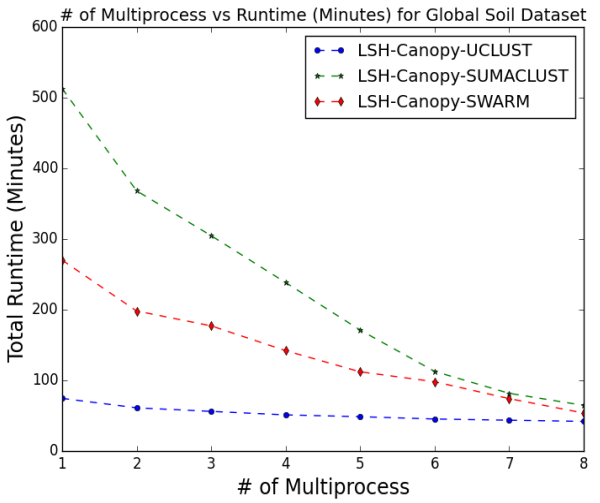
(c)



(d)



(e)



(f)

Figure 4. **Runtime comparison of LSH-Canopy framework with UCLUST, SUMACLUST and SWARM on Bokulich_2, Bokulich_3, Bokulich_6, Body Sites, Canadian Soil and GLobal Soil datasets.**