HTSFinder: Powerful Pipeline of DNA Signature Discovery by Parallel and Distributed Computing

Ramin Karimi[1] and Andras Hajdu[1,2]

[1]Faculty of Informatics, Department of Computer Graphics and Image Processing, University of Debrecen, Debrecen, Hungary.
[2]Bioinformatics Research Group, University of Debrecen, Debrecen, Hungary.

ABSTRACT: Comprehensive effort for low-cost sequencing in the past few years has led to the growth of complete genome databases. In parallel with this effort, a strong need, fast and cost-effective methods and applications have been developed to accelerate sequence analysis. Identification is the very first step of this task. Due to the difficulties, high costs, and computational challenges of alignment-based approaches, an alternative universal identification method is highly required. Like an alignment-free approach, DNA signatures have provided new opportunities for the rapid identification of species. In this paper, we present an effective pipeline HTSFinder (high-throughput signature finder) with a corresponding *k*-mer generator GkmerG (genome *k*-mers generator). Using this pipeline, we determine the frequency of *k*-mers from the available complete genome databases for the detection of extensive DNA signatures in a reasonably short time. Our application can detect both unique and common signatures in the arbitrarily selected target and nontarget databases. Hadoop and MapReduce as parallel and distributed computing tools with commodity hardware are used in this pipeline. This approach brings the power of high-performance computing into the ordinary desktop personal computers for discovering DNA signatures in large databases such as bacterial genome. A considerable number of detected unique and common DNA signatures of the target database bring the opportunities to improve the identification process not only for polymerase chain reaction and microarray assays but also for more complex scenarios such as metagenomics and next-generation sequencing analysis.

KEYWORDS: DNA signature, *k*-mers, Hadoop, WordCount, MapReduce, Hive

CITATION: Karimi and Hajdu. HTSFinder: Powerful Pipeline of DNA Signature Discovery by Parallel and Distributed Computing. *Evolutionary Bioinformatics* 2016:12 73–85 doi: 10.4137/EBO.S35545.

TYPE: Technical Advance

RECEIVED: September 28, 2015. **RESUBMITTED:** November 5, 2015. **ACCEPTED FOR PUBLICATION:** December 05, 2015.

ACADEMIC EDITOR: Jike Cui, Associate Editor

PEER REVIEW: Six peer reviewers contributed to the peer review report. Reviewers' reports totaled 2604 words, excluding any confidential comments to the academic editor.

FUNDING: This study was supported in part by the Project TAMOP-4.2.2.C-11/1/KONV-2012-0001 supported by the European Union and cofinanced by the European Social Fund. The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: raminkm2000@yahoo.ca

Paper subject to independent expert blind peer review. All editorial decisions made by independent academic editor. Upon submission manuscript was subject to antiplagiarism scanning. Prior to publication all authors have given signed confirmation of agreement to article publication and compliance with all applicable ethical and legal requirements, including the accuracy of author and contributor information, disclosure of competing interests and funding sources, compliance with ethical requirements relating to human and animal study participants, and compliance with any copyright requirements of third parties. This journal is a member of the Committee on Publication Ethics (COPE).

Published by Libertas Academica. Learn more about this journal.

Introduction

DNA signature is a short *k*-mer oligonucleotide fragment with an arbitrary length *k*, which is unique or specific for a particular group of species selected from a target genome database. There are two categories of unique and common signatures according to the purpose of usage. The presence of a unique DNA signature in any volume of sequences and genetic materials represents the existence of the corresponding species.[1,2] Therefore, signature discovery is the action of finding specific fragments of genome in a database.[3] Any pipeline, application, or algorithm that is designed for DNA signature discovery has to detect an entire database or multiple databases recursively. The procedure varies according to the purpose of using DNA signatures.

Despite the impact of the sequences 16S rDNA and 16S rRNA in the microbial taxonomy, they are particularly useful for taxa above the rank of species. Because of sequence similarities, they are not sufficient to define bacterial species and strains.[4] Approximately 15% of bacterial genomes contain only a single copy of 16S rRNA.[5] Since the high-throughput sequences are often noisy and partial,[6] the application of 16S sequences for the next-generation sequencing (NGS) data analysis at species level is even less efficient. Concerning the large number of DNA signatures in different species and the possibility to choose arbitrary lengths of them for identification, this approach is not only suitable for polymerase chain reaction (PCR) and microarray-based assays but also has great potential for NGS analysis. The pipeline high-throughput signature finder (HTSFinder) that is proposed in this paper has been designed to address some of the challenges of DNA signature discovery in order to enhance the usability of DNA signatures for NGS analysis.

Several tools and algorithms of DNA signature discovery have been proposed in the literature in order to facilitate the design of microbial and pathogen-based diagnostic assays; notable instances are discussed in the following sections.

Tool for Oligonucleotide Fingerprint Identification (TOFI)[7] is designed to identify DNA fingerprints of a single genome as suitable probes for microarray-based diagnostic assays. It utilizes the whole genome of the pathogen instead of the special gene (such as 16s rRNA) or special regions of the genome for designing probes.[8] In order to design DNA microarray probes, TOFI reduces the solution space by discarding DNA sequences that are common to the target sequence and one or more phylogenetically close sequences. Then, each extracted DNA microarray probe is compared with all DNA sequences from the chosen reference database.[7]

Tool for PCR Signature Identification (TOPSI)[9] is a pipeline for real-time PCR signature discovery. TOPSI detects common signatures among multiple strains of bacterial genomes by collecting the shared regions through pairwise alignments between the input genomes. It is an extended version of TOFI.[9,10]

Insignia[11] provides unique signatures that can be used to design primers for PCR and probes for microarray assays. It has two main components: the web interface and the computational pipeline. The computational pipeline uses grid computing and an algorithm to perform pairwise alignment of every pair of target genomes and background genomes for their comparison. Insignia provides signatures that are unique against the background genomes based on databases of bacterial and viral genomic sequences containing 13,928 organisms (11,274 viruses/phages and 2,653 bacteria).[12] In fact, when a user adjusts the desired options in the Insignia web interface, a query runs on the database that contains the results of DNA signature discovery which has already been provided.

TOFI, TOPSI, and Insignia use the open-source software MUMmer[13] that implements a suffix-tree-based algorithm for comparing genomic sequences.[7,9,11] It is a package for the alignment of very large DNA and amino acid sequences. Furthermore, these three pipelines use Basic Local Alignment Search Tool (BLAST) for the evaluation of signatures regarding specificity.

CaSSiS[14] is an algorithm for detecting signatures with maximal group coverage within a user-defined specificity range for designing primers and probes. It provides signatures for single or group organisms in hierarchically clustered sequence datasets. This algorithm calculates the Hamming distance between a signature candidate and its matched targets. CaSSiS uses the rRNA sequences provided by the database SILVA to create a signature collection for designing primers and probes.

The consecutive multiple discovery (CMD) algorithm[15] is an iterative method including the parallel and incremental signature discovery (PISD) method as a kernel routine to discover implicit DNA signatures. PISD is a combination of the Hamming-distance-based algorithm, the Internal-memory-based unique signature discovery (IMUS) approach,[16] and Zheng's method[17] in terms of using the corresponding incremental and parallel computing techniques. PISD uses a mismatch tolerance and previously discovered signatures of specific lengths as candidates to find shorter signatures instead of scanning the whole database. CMD and PISD can find unique signatures for single sequences, but cannot search for signatures that are specific for groups;[16] they are designed to find signatures of sequences from expressed sequence tag (EST) databases.

The internal memory-based unique signature discovery algorithm IMUS[16] is an improvement of Zheng's method,[17] which is based on the Hamming distance for detecting unique signatures. IMUS tries to discard similar substrings of a sequence in order to obtain the DNA signatures as unique fragments. Parallel internal-memory-based unique signature discovery (PIMUS)[18] is the improved version of IMUS. Both algorithms load the complete DNA database into the main memory to find unique signatures in EST datasets.

Distributed divide-and-conquer-based signature discovery (DDCSD)[19] applies a divide-and-conquer strategy for detecting DNA signatures. When the dataset is large and cannot be loaded into the memory all at once, the algorithm splits it into smaller segments in which parts are loaded and processed one by one. The discovery node and the discovery routine are the main components of this algorithm. When the size of the dataset is larger than the available memory, the discovery routine splits the dataset into multiple parts that are processed one at a time by the discovery nodes. This algorithm is based on searching for similarities and mismatches in the patterns. Similar to CMD, PISD, IMUS, and PIMUS, this algorithm is designed to search EST datasets, but it can process larger databases such as the human whole-genome EST database as well. Table 1 contains some more details on the algorithms described earlier.

Jellyfish[20] is an algorithm to count the *k*-mers in parallel. This algorithm implements a lock-free hash table optimization for counting *k*-mers up to 31 bases in length.

There are other approaches to find signatures or probe sequences, such as PROBESEL,[21] OligoArray,[22] OligoWiz,[23] YODA,[24] PRIMROSE,[25] and ARB-ProbeDesign.[26] All of them are limited to one selected target or single sequence in each run; thus, they are not applicable for large datasets.[14]

In practice, despite the respected efforts of abovementioned and other methods, there are still a number of limitations for DNA signature discovery.

Since most existing methods of DNA signature discovery require significant computational resources, they are not applicable for the entire research community. Due to the size of genome databases, the large amount of random-access memory (RAM) and central processing unit (CPU) capacity requirements and long execution times are the major limitations of most of the abovementioned methods that are based on pattern comparison and pairwise alignment of the genomes. The determination of the mismatch tolerance level as a discovery condition also influences the results.

In some cases, it is necessary to load the whole dataset into the main memory for searching for unique or common signatures. When the size of the data exceeds the available memory, the execution will fail. For instance, in IMUS, PIMUS, and Zheng's methods, the entire database has to be loaded into the memory.[19] Thus, for such sequential algorithms like IMUS, increasing the number of CPU cores does not increase the discovery efficiency of the algorithm.[18] Another limitation for

Table 1. A comparison of signature discovery algorithms according to the data format, computational resources, and ability to process single or multiple sequences.

NAME	DATA FORMAT	ADOPTED PLATFORM ACCORDING TO THE PUBLICATION	BLAST SPECIFICITY	ABILITY FOR SINGLE SEQUENCE	ABILITY FOR MULTIPLE SEQUENCES
TOFI	FASTA	64 x 1.5 GHz Itanium 2 processors running on Linux with 64 GB of shared memory	✓	✓	×
TOPSI	FASTA	98-cores Linux cluster	✓	✓	✓
Insignia	FASTA	192-node Linux cluster	✓	✓	✓
CaSSiS	rRNA	Intel Core i7 CPU (4 cores, 2.67 GHz) with 24 GB of RAM	×	✓	✓
CMD and PISD	ESTs	Dell PowerEdge R900 server with two Intel Xeon E7430 2.13 GHz quad-core CPUs, 12 GB RAM	×	✓	×
IMUS	ESTs	Intel 2.93GHz CPU	×	✓	×
PIMUS	ESTs	Intel Core i7 870 2.93GHz quad-core CPU and 16 GB RAM	×	✓	×
DDCSD	ESTs	A Master node: Intel Core i7 CPU 870 at 2.93 GHz and 16 GB RAM 10 Slave nodes: Intel Core i7 CPU 3770 K at 3.50 GHz and 32 GB of RAM for each one	×	✓	✓

most of the abovementioned methods is the lack of efficiency to find both unique and common signatures simultaneously. Most of them are capable to find only DNA signatures of a single genome. In addition, the limitation of some of these methods is the lack of the possibility to select an arbitrary length (*k*) for the signatures.

The additional challenge as another major limitation for DNA signature discovery methods is the lack of option in the choice of target and nontarget genome databases. TOFI, TOPSI, and Insignia use BLAST databases (such as nt and nr databases) for the background or nontarget genomes for specificity evaluation of signatures and there is no option for the user to choose other target and nontarget genome databases. As an example, in the Insignia web interface, the user receives a quick response without special requirements on local computational resources. However, this privilege comes with the restriction that there is no option to use other sequences as the target and background genomes, because they are part of the Insignia database.[19] With the advancements of the sequencing technologies and the increasing number of complete genomes, whole-genome shotgun sequences, and draft genomes, it is obvious that some of these signatures will not be unique later using BLAST specificity evaluation. This issue is a challenge not only for DNA signatures but also for all the sequence-based identification methods.

Geographical distribution and diversity of the species, ecological and chemical status, host and environmental factors, isolation or complexity of the samples, and many other factors can have a great impact on the selection of target and nontarget genome databases for DNA signature discovery. When the absence of a considerable number of species in the sample is evident, it seems quite questionable that we eliminate a large number of useful DNA signatures through their assessment and specificity evaluation against the entire background sequence databases such as BLAST. For instance, when we are sure that, in the sample, there is nothing from zebra fish, mouse, chimpanzee, black cottonwood, *Macaca fascicularis*, etc., we do not need to check the uniqueness of our DNA signatures against their genomes; otherwise, we would lose a significant number of signatures.

ESTs are short fragments of mRNA sequences obtained by single sequencing of randomly selected cDNA clones. ESTs are mostly used either to identify gene transcripts or as an alternative cheap method of gene discovery and gene sequence determination.[27]

IMUS, PIMUS, CMD, PISD, and DDCSD are designed to scan EST sequences for the unique signatures. However, the ESTs represent only fragments of genes, not complete coding sequences;[28] therefore, many signatures are missed.

The pipeline HTSFinder has significant advantages compared with the DNA signature discovery pipelines and algorithms described earlier.

- First, HTSFinder is capable to detect all unique, common, and maximal group coverage signatures of the entire database or multiple databases simultaneously.
- Second, it becomes possible to select target and nontarget genome databases, based on user requirements. For instance, we have the ability to use both forward and reverse-complement genome sequences of a database for detecting DNA signatures.
- Third, the pipeline can be considered either a cluster of low-cost computer nodes that are commonly available in research facilities or a high-performance computing (HPC).

- Finally, the flexibility of the different phases of the pipeline makes it suitable for other bioinformatic and metagenomic studies such as NGS analysis.

HTSFinder is very efficient and powerful with high accuracy for both unique and group-specific signatures without discarding even a single signature from the database, except those ones that contain the International Union of Pure and Applied Chemistry (IUPAC) nucleotide codes such as K, M, N, R, S, W, and Y. Our GkmerG component will remove any *k*-mer containing IUPAC nucleotide codes after generating the *k*-mers. In this pipeline, there is nothing to worry about the mismatch tolerance and complexity of comparison and pairwise alignment search methods.

Materials and Methods

Description of the pipeline. HTSFinder consists of three computational phases as shown in Figure 1. This pipeline generates all the possibilities of *k*-mers for every genome individually and then determines their frequency in the entire database. Finally, DNA signatures of every species or strain are obtained in the database or multiple databases that have been involved in the pipeline. HTSFinder implements the parallel and distributed computational tool Hadoop for the second and third phases.

Data preparation. The first phase of the pipeline is carried out by GkmerG that is designed to obtain all the possibilities of *k*-mers of genome sequences with FAST-All (FASTA) format (*.fna or *.fa). This software tool removes the remarks of the genome and splits it to the specific length *k*. Then, it eliminates the *k*-mers that contain IUPAC nucleotide codes and every subsequence of length less than *k* which has remained from the end of the sequence after splitting. Figure 2 illustrates the split of the genome by GkmerG. Concatenating the files, sorting *k*-mers, and removing all duplicates except one are the last steps of GkmerG. For the species with multiple chromosomes and some bacterial genomes that are composed of multiple chromosomes[29] and plasmids, GkmerG concatenates them into a single file before sorting at the end of the first phase. GkmerG copies the original database into another directory as the reference database by appending a number to the beginning of every species name in it, to simplify the future data management. Once we get the output of the first phase for a database, we can keep it forever. In case of any update in the database, we need only to repeat this phase for the updated or new genomes, not for the whole database. The output of GkmerG is the input for the second step in the pipeline which is described in the following section.

The Apache Hadoop. Dramatic increase in the amount of data in various, particularly biological fields of science revealed the inadequacy of existing ordinary computers for big data analytics. It has prompted the developers to compose tools and applications using parallel and distributed computing that could be applicable on commodity hardware. The Apache Hadoop project[30–32] has been designed as an open source, Java-based software framework for parallel and distributed computing on large datasets using commodity hardware. Hadoop allows to run simple programming models on large structured and unstructured datasets across an arbitrary number of nodes in a cluster. A Hadoop cluster has single master and several slave nodes that are connected to each other through Secure Shell. It can run as a single-node or multinode cluster with thousands of nodes. The Hadoop core has two primary components: Hadoop Distributed File System (HDFS) and MapReduce.

HDFS[31,33] is the data storage part of Hadoop. It provides high-throughput access to large datasets across multinodes of a cluster. HDFS breaks down the data into small chunks, which are stored as independent elements. HDFS provides input data storage for the MapReduce framework.

MapReduce[31] is a programming model for parallel and distributed data processing. MapReduce works by breaking

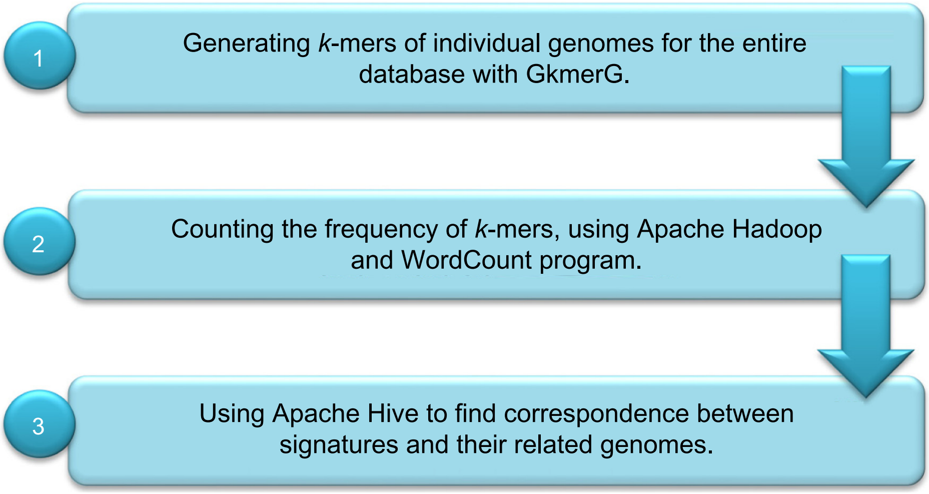
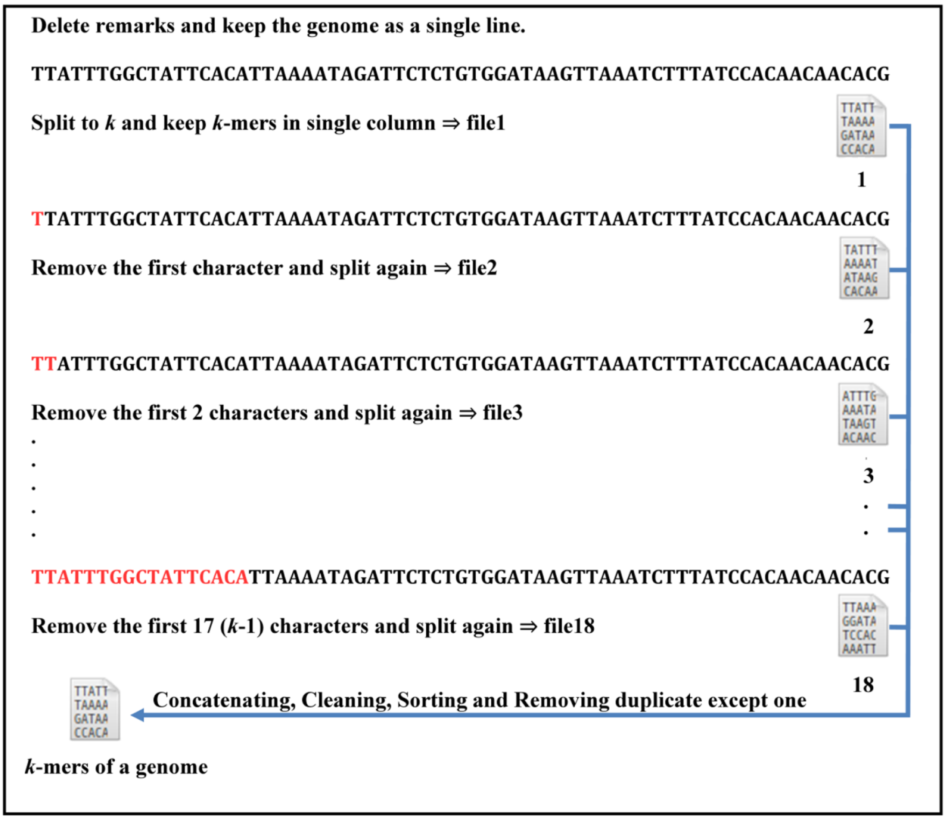

1
Generating *k*-mers of individual genomes for the entire database with GkmerG.
2
Counting the frequency of *k*-mers, using Apache Hadoop and WordCount program.
3
Using Apache Hive to find correspondence between signatures and their related genomes.

Figure 1. The three main phases of HTSFinder for detecting DNA signatures. We can repeat the second phase with the obtained results if required.



Delete remarks and keep the genome as a single line.

TTATTTGGCTATTCACATTAAAATAGATTCTCTGTGGATAAGTTAAATCTTTATCCACAACAACACG

Split to *k* and keep *k*-mers in single column ⇒ file1

TTATTTGGCTATTCACATTAAAATAGATTCTCTGTGGATAAGTTAAATCTTTATCCACAACAACACG

Remove the first character and split again ⇒ file2

TTATTTGGCTATTCACATTAAAATAGATTCTCTGTGGATAAGTTAAATCTTTATCCACAACAACACG

Remove the first 2 characters and split again ⇒ file3

TTATTTGGCTATTCACATTAAAATAGATTCTCTGTGGATAAGTTAAATCTTTATCCACAACAACACG

Remove the first 17 (*k*-1) characters and split again ⇒ file18

Concatenating, Cleaning, Sorting and Removing duplicate except one

k-mers of a genome

Figure 2. Splitting of the genome by GkmerG for $k = 18$ to get all the possibilities of 18-mers. Generating *k*-mers for a single genome with GkmerG includes: purgation, splitting, concatenation, cleaning, sorting, and removing duplicate except one. The output of GkmerG is a file containing *k*-mers of a genome in a single column. The labels above the file numbers in this figure represent the beginning of four *k*-mers in the head of files.

the processing into two phases: the Map and the Reduce. The Map phase processes a set of data in parallel and returns it as an intermediate result, and then the Reduce phase reduces it to a smaller set of data. Each Map and Reduce works independently. In fact, MapReduce decreases the large amount of raw input data into smaller amount of useful data for further processing.[34,35]

As another component of the second-generation Hadoop 2 release of Apache, YARN (Yet Another Resource Negotiator) was added in order to upgrade scheduling, resource management, and execution in Hadoop.[36]

Although Hadoop is defined as a distributed system with multi-nodes, the ability of Apache Hadoop to use MapReduce for parallel processing of large datasets is an extra power to let even a single-node processes large datasets exceeding memory and CPU capacity.

In this research, we used the Hadoop framework and WordCount program to calculate the frequency of *k*-mers in very large genome datasets. In Hadoop 1.2.1 and earlier releases, the JAR (Java Archive) file of WordCount is also included. Figure 3 illustrates a MapReduce and WordCount process.

In the second step of the pipeline, we copy all the output files of the first step to the HDFS and run the WordCount program.

The result of this step is a large file containing sorted and non-duplicate list of *k*-mers obtained from the files generated in the first step in one column and another column containing the frequencies of *k*-mers among genomes of the database. A *k*-mer with a frequency value 1 indicates that this *k*-mer is a unique substring that appeared only in one of the species in the database. These occurrences are primarily what we are looking for as unique DNA signatures. Any value preceding a *k*-mer indicates the number of genomes (species) that contain the given *k*-mer. Table 2 shows a portion of the Hadoop and WordCount output. For instance, the 18-mer with frequency 8 in the first row of Table 2 means that this 18-mer occurs in eight genomes among the 2,773 bacterial ones, while the 18-mer in the fourth row is a unique signature in the database. In the second step of the pipeline, we can extract all unique signatures or group-specific signatures due to the frequency, but we cannot determine the owner of the signatures.

Once we execute the second phase for a database, we can use the results in the future until the next update of the database. However, as a difference from the updatable first phase, in case of any update in the database, we have to repeat the second phase for the entire database.

When there are multiple target and nontarget databases, it is possible to merge all of them in the pipeline, but

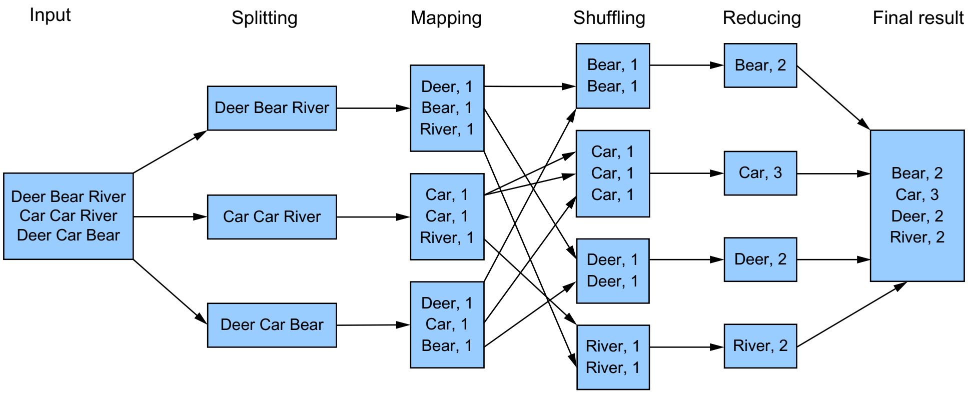


Figure 3. An example of the overall MapReduce WordCount process. The original image was made by Trifork.

as the input grows larger, it requires far more computational resources. As a suggestion, it is better to implement the first and second steps for every database separately. With respect to the WordCount function that discards repeated *k*-mers and keeps only one in the output, we can reduce the size of output files and also the execution time. Then, we can merge the output of the second phase for all the databases and repeat the second phase with WordCount in Hadoop one more time. In this case, we have a shorter process. Moreover, for the future execution, we can select the output files of the second phase as the candidate of their corresponding databases. In this case, we do not need to perform the first phase of the pipeline for the previously processed databases and we can repeat the second phase for target and nontarget databases by merging the smaller files. For instance, the output of the first phase for the bacterial genome database resulted in a file with 177.35 GB of 18-mers. However, in the second phase, the size of this file was reduced to 103.03 GB that contained all the candidates of 18-mers in the database without any repeat. We can use this file as the candidate of bacterial genome database for further processing. Figure 4 illustrates the process of finding DNA signatures of the target database among nontarget databases.

Table 2. An example of Hadoop and WordCount results.

SIGNATURES OR 18-MERS	FREQUENCY IN THE DATABASE
AAAAAAAAAAAAAAAGAG	8
AAAAAAAAAAAAAAAGAT	25
AAAAAAAAAAAAAAAGCA	20
AAAAAAAAAAAAAAAGCC	1
AAAAAAAAAAAAAAAGCG	5
AAAAAAAAAAAAAAAGCT	6
AAAAAAAAAAAAAAAGGA	9
AAAAAAAAAAAAAAAGGC	3
AAAAAAAAAAAAAAAGGG	6
AAAAAAAAAAAAAAAGGT	38

The input for the third phase of the pipeline is the output of the first and second phases. The proper steps of the third phase are described in the following section.

The Apache Hive. *Hive*[37–39] is a data warehouse infrastructure on the top of the Hadoop MapReduce framework. It is designed to query a large dataset that is stored in the HDFS using an SQL-like language called HiveQL. Traditional, relational databases require the data to be in a structured format, while Hive can handle both structured and unstructured information. It lets the user to process large datasets with relatively little effort and in a reasonably short time. This research proves the efficiency of Hive to handle querying on billions of rows in a table or multiple tables. With HiveQL, we can extract whatever we need from the results of the second step of the pipeline. We can extract all the unique signatures of a specific species in the database or group-specific signatures that are common among 2, 3, 4, etc. Due to the flexibility of querying in Hive, there are various ways to create the tables and design the queries in the third step. Our future study is motivated by optimizing querying. After loading the data into the tables created with Hive, we can use queries such as SELECT and JOIN to extract relationships. We should create two tables with Hive: one for the output of the first phase and another one for the complete or a special part of the output of the second phase. By considering the ability of Hive to query very large tables and prevent the repetition of queries, we added a column containing the reference number to the files from the first step. For example, file 1 contains 18-mers from the first species in the database, so we inserted a column containing reference index 1 before all 18-mers in this file. Then, we merged all the 2,773 files in a single large one (220.35 GB) with two columns of *k*-mers and their related reference numbers. The reference number indicates the number that has been appended to the name of the species by GkmerG in the first phase.

There are several options to create the table from the output of the second step: one is to create the table without making any changes in the output and another one is to break down the output into smaller groups according to the targeted

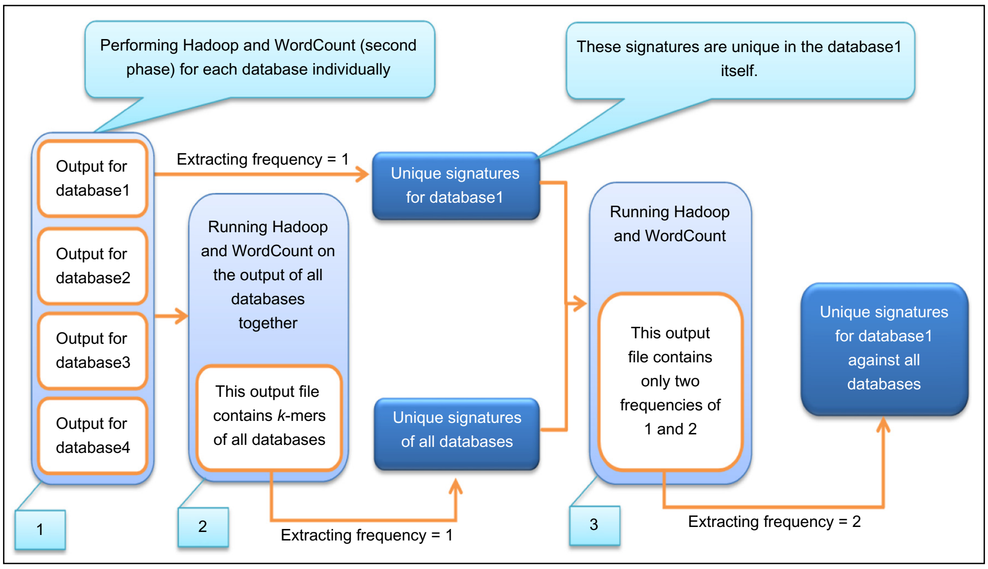

Performing Hadoop and WordCount (second phase) for each database individually
These signatures are unique in the database1 itself.
Output for database1
Output for database2
Output for database3
Output for database4
Extracting frequency = 1
Unique signatures for database1
Running Hadoop and WordCount on the output of all databases together
This output file contains k-mers of all databases
Unique signatures of all databases
Running Hadoop and WordCount
This output file contains only two frequencies of 1 and 2
Unique signatures for database1 against all databases
1
2
Extracting frequency = 1
3
Extracting frequency = 2

Figure 4. The recommended process for detecting unique DNA signatures of a target database against nontarget databases. In step 2 of this figure, the frequency number of *k*-mers varies from 1 to *n*, where *n* is the total number of the databases that are used in the pipeline. Since there are four databases in this figure, the frequency of *k*-mers in step 2 is from 1 to 4. In step 3, there are two input files with the list of non-repeated *k*-mers; therefore, the frequency of *k*-mers in the output is 1 or 2. Hence, *k*-mers with frequency 2 that is common in both input files are the unique signatures of database 1 against all databases.

signature. For example, if we are looking for the unique signatures, it would be better to extract only 18-mers with frequency 1. However, if we are looking for a common signature, then it would be better to extract the 18-mers with a specific frequency number such as 2, 3, 4, etc. In order to have a faster and easier implementation with Hive and later steps, we recommend the second option.

Source code for GkmerG and information and command lines for Hadoop and Hive are freely available at: http://www.inf.unideb.hu/~hajdua/HTSFinder.html, https://sourceforge.net/projects/htsfinder/, and https://github.com/raminkm/HTSFinder.

Selected sequence databases. *Bacterial genome database.* To prove the efficiency of our proposed method, the bacterial genome databases with 2,773 complete genomes in FASTA format (*.fna) were downloaded from the National Center for Biotechnology Information (NCBI) database. The size of this database is 9.7 GB after decompression. The list of the bacterial genomes is available in Supplementary Files.

The reverse-complement bacterial genome database. Another database that we used in this study was the reverse-complement bacterial genome database. The revcom.pl 1.2 (available at: http://code.google.com/p/nash-bioinformatics-codelets/) is a Perl program written by John Nash (Copyright © Government of Canada, 2000–2012). We used this program to provide the reverse-complement sequences for the whole bacterial genome database.

Human genome database. The whole human genome is another database used in this research. The Homo sapiens *hs-ref-GRCh*38 sequences in FASTA format (*.fa.gz) were downloaded from the NCBI ftp database. The size of the genome was 2.9 GB after decompression.

Results and Discussion

Results for the bacterial genome database. GkmerG has generated 2,773 files with a total size of 177.35 GB containing all the possibilities of 18-mers from individual bacterial genomes in the first step of the pipeline. After copying the results of the first step into the HDFS, we ran the WordCount program using Hadoop in order to determine the frequencies of the 18-mers in the 2,773 files. The result of this execution was a 103.03-GB file with two columns. The first column contained the 18-mers or signatures, while the second one contains the frequency number of each 18-mer. Frequency 1 in this file represents the uniqueness of the related

Table 3. Total number of 10 least and 10 most common signatures in the bacterial genome database.

FREQUENCY (LEAST COMMON)	NUMBER OF SIGNATURES IN THE DATABASE	FREQUENCY (MOST COMMON)	NUMBER OF 18-MERS IN THE DATABASE
1	3,552,866,254	2040	1
2	689,790,798	2042	1
3	245,109,794	2044	1
4	114,234,398	2074	2
5	68,395,645	2075	1
6	48,107,467	2102	1
7	31,544,271	2112	1
8	26,164,511	2113	2
9	23,650,821	2114	2
10	16,156,541	2125	1

Table 4. An example of the output for the third phase (the right side of the table). The reference numbers in this table indicates the numbers appended by GkmerG for easier tracking of data in the pipeline.

SIGNATURE	GkmerG REFERENCE NUMBER	NAME OF THE BACTERIAL GENOME
AAAAACGCTCTGATATGA	1059	*Eubacterium_rectale_ATCC_33656_uid59169*
AAAAACGCTCTGCCACCA	1520	*Methanobacterium_SWAN_1_uid67359*
AAAAACGCTCTGGGAATT	705	*Chromohalobacter_salexigens_DSM_3043_uid62921*
AAAAACGCTCTTTTATTT	472	*Campylobacter_hominis_ATCC_BAA_381_uid58981*
AAAAACGCTGAAACGCCT	2649	*Tolumonas_auensis_DSM_9187_uid59395*
AAAAACGCTGAAATCCGC	2013	*Rahnella_Y9602_uid62715*
AAAAACGCTGAATGAAGC	39	*Acinetobacter_ADP1_uid61597*
AAAAACGCTGACAATAAA	1337	*Lactobacillus_brevis_KB290_uid195560*
AAAAACGCTGACCTTCTA	1	*Acaryochloris_marina_MBIC11017_uid58167*
AAAAACGCTGACGGAAGT	2126	*Ruminococcus_albus_7_uid51721*

signature in the entire database. In other words, an 18-mer with frequency 1 is a unique signature among 2,773 bacterial genomes and an 18-mer with frequency 2 is a common signature which is presented in two genomes among 2,773 bacterial genomes. Table 3 represents the quantity of 10 least common and 10 most common 18-mers with their frequencies in the bacterial genome databases. This table shows that 3,552,866,254 of signatures are unique in the database and there is one subsequence (18-mer) that is repeated in 2,125 bacterial genomes.

In the third phase, we have created tables in Hive and loaded files from the first and second phases. The table with the reference numbers and k-mers (220.35 GB) and the table with the list of unique signatures (67.5 GB) are used to run the query in Hive in order to specify the species and strains as the owners of the unique signatures. We have repeated the query on the table containing the list of signatures with frequency 2 instead of the unique signature's table to find every pair of species with a common signature. For other frequencies, the same implementation is required.

As shown in Table 4, the output of the third phase was a file with two columns containing the following: the signature and the reference number indicating its corresponding bacterial genome in the reference database created by GkmerG in the first phase.

The following examples are parts of the results obtained by HTSFinder to show the efficiency of this pipeline.

No unique DNA signatures with $k = 18$ were found for 30 of the bacterial genomes in the database. They are listed in Supplementary Files.

The number of unique DNA signatures in 475 genomes was <10,000. *Chlamydia* as a genus of bacteria with 83 species

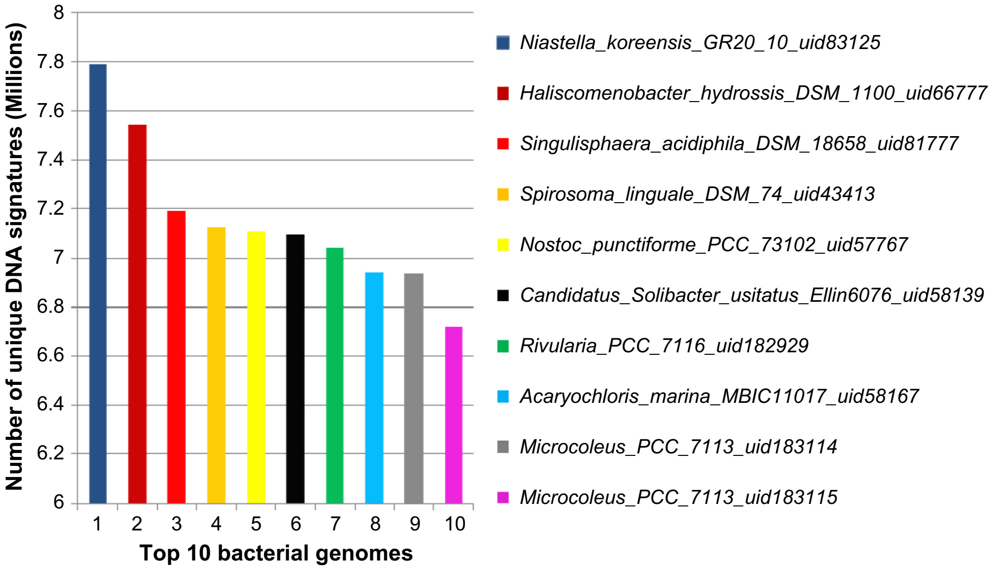


Figure 5. Top 10 bacterial genomes with the highest number of unique DNA signatures in the bacterial genome database.

Table 5 *B. mallei* and *B. pseudomallei* genomes with their number of unique DNA signatures of 18-mers in the bacterial genome database.

THE REFERENCE NUMBER AND NAME OF THE BURKHOLDERIA GENOMES	NUMBER OF UNIQUE DNA SIGNATURES
Burkholderia_mallei_ATCC_23344_uid57725	90,278
Burkholderia_mallei_NCTC_10229_uid58383	24,858
Burkholderia_mallei_NCTC_10247_uid58385	19,442
Burkholderia_mallei_SAVP1_uid58387	7,649
Burkholderia_pseudomallei_1026b_uid162511	282,992
Burkholderia_pseudomallei_1106a_uid58515	173,688
Burkholderia_pseudomallei_1710b_uid58391	41,153
Burkholderia_pseudomallei_668_uid58389	218,985
Burkholderia_pseudomallei_BPC006_uid174460	81,768
Burkholderia_pseudomallei_K96243_uid57733	195,711
Burkholderia_pseudomallei_MSHR305_uid213227	320,198
Burkholderia_pseudomallei_MSHR346_uid55259	172,551
Burkholderia_pseudomallei_NCTC_13179_uid226109	382,494

and strains in the bacterial genome database has the lowest number of unique DNA signatures of 18-mers. The number of the unique signatures of 18-mers in 75 of them was <10,000 and in 57 it was <1,000. We have located 13 *Chlamydia* bacteria without unique signatures with $k = 18$. Top 10 bacterial genomes with the highest number of unique DNA signatures in the bacterial genome database are shown in Figure 5.

Burkholderia mallei and *Burkholderia pseudomallei* are two closely related pathogens that are very difficult cases for PCR assays. These two bacteria are the causative agents of glanders and melioidosis diseases in humans and animals.[9,40,41] Due to the phenotypic and genotypic similarity of them, until a few years ago, they were considered to have the same species status. Concerning the literature, only one PCR signature was reported to be unique to *B. mallei*.[9,40] The HTSFinder pipeline could detect a considerable number of DNA signatures for *B. mallei* and *B. pseudomallei*. Although these signatures are just unique in the bacterial genome database, due to the notable number of signatures listed in Table 5, it is evident that under different circumstances it would be better to have an alternative opportunity to define the uniqueness of the DNA signatures and to select the target databases according to the requirements. Moreover, it should be noted that much more DNA signatures could be found by increasing the length of *k*-mers.

For the frequencies >1, this pipeline detects the common signatures not only among a single species and its strains but also in the entire database. Frequencies 2 and 3 have been considered samples to prove the efficiency of this pipeline for discovering common DNA signatures within the bacterial genome database.

As an example, the following results were obtained for *Acaryochloris_marina_MBIC11017_uid58167* that is the first bacteria in the database.

A total of 689,790,798 signatures of $k = 18$ with frequency 2 were found in the database, whereas 673,490 of them are shared between *Acaryochloris_marina_MBIC11017_uid58167* and 2,382 other species.

There was not any signature of $k = 18$ with frequency 2 between *Acaryochloris_marina_MBIC11017_uid58167* and 390 other bacterial genomes. Figure 6 presents the highest number

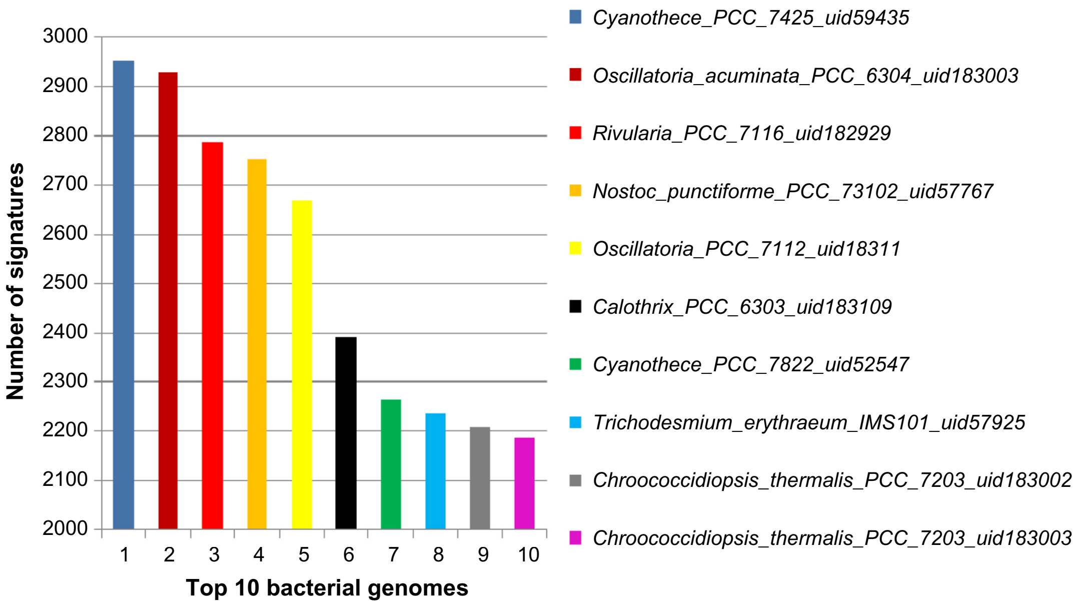


Figure 6. Ten bacterial genomes with the highest number of signatures common with *Acaryochloris_marina_MBIC11017_uid58167* in the bacterial genomes database. This is an example of the results for common signatures with frequency 2 obtained by HTSFinder.

Table 6. A portion of results for signatures with frequencies 2 and 3 in the database. Concerning the reference numbers, most of the common signatures are shared among the phylogenetically close genomes. However, number of common signatures among unrelated species are also notable.

SIGNATURES WITH FREQUENCY = 2	GkmerG REFERENCE NUMBERS		SIGNATURES WITH FREQUENCY = 3	GkmerG REFERENCE NUMBERS		
AAAAAAAAAAGATAAATA	355	508	AAAAAAAAAAAAATATCG	1709	1708	2677
AAAAAAAAACAGACACAA	2110	2109	AAAAAAAAAAAACAGAAC	1249	1255	1267
AAAAAAAAACAGCATTAA	2209	2214	AAAAAAAAAATAAATACA	2726	2734	542
AAAAAAAAACAGGCTTAC	394	1499	AAAAAAAAAGAAACAAAG	681	678	679
AAAAAAAAACCGCCGAAC	1046	1048	AAAAAAAAAGATGTTAAT	969	2384	247
AAAAAAAAACCGCTTTTA	1879	1265	AAAAAAAAAGCAAAACAA	2223	355	102
AAAAAAAAACGAACAAAC	101	1813	AAAAAAAAAGTAAATGCG	1793	2731	2730
AAAAAAAAACGATTCAGA	2106	2107	AAAAAAAAATAGACAATG	498	500	755
AAAAAAAAACTAATGCTT	349	355	AAAAAAAAATATTCATGC	321	897	560
AAAAAAAAACTAATTCTG	1406	1408	AAAAAAAAATTCAAAATT	567	505	325
AAAAAAAAAGAACCAAAC	544	545	AAAAAAAAATTTAGCGAT	2714	1814	321
AAAAAAAAAGACTGACTC	2696	1066	AAAAAAAAATTTTTATAG	703	402	324
AAAAAAAAAGATGTTGTA	545	544	AAAAAAAACAAGAAGCGC	1426	1427	1428
AAAAAAAAAGGATTCGAA	1428	1427	AAAAAAAACAATTAGCGA	1128	2677	2404
AAAAAAAAATAAAGACTC	345	343	AAAAAAAACAGATAGTGA	2115	1061	1508
AAAAAAAAATAGTGACGA	1686	1693	AAAAAAAACAGCAGCACC	2535	1584	1058

of signatures with frequency 2 which are common between *Acaryochloris_marina_MBIC11017_uid58167* and 10 other bacterial genomes in the database.

The results of the executions for signatures with frequencies 2 and 3 showed that most of the signatures are shared among phylogenetically close species of the database. However, there were also a lot of signatures that belonged to unrelated bacterial genera and families. Table 6 shows a partial view of the results for frequencies 2 and 3. From 245,109,794 signatures of frequency 3 in the database, 160,264 of them are shared between *Acaryochloris_marina_MBIC11017_uid58167* and two other species.

A series of lengths *k* from 21 to 30 have been considered to evaluate the effect of increasing the length of the signature in the results and to compare the results of the odd and the even number of *k*. Table 7 contains the number of unique DNA signatures for the human genome and three chromosomes 1, *x*, and *y*, which represent large, medium, and small sequences in the human genome. This table shows that increasing the length of the signature causes increasing the number of unique signatures and there is not a meaningful difference between the odd and the even numbers.

To compare the number of unique signatures against the within-species variability and the entire bacterial genome

Table 7. Number of unique DNA signatures in the human genome and its three chromosomes with different sequence sizes for a series of lengths of *k*-mers from 21 to 30.

LENGTH OF SIGNATURE	THE WHOLE GENOME (2.8 GB)	CHR1 (222 MB)	CHRX (147 MB)	CHRY (19 MB)
k = 21	2.24297e+09	176,137,004	109,691,126	10,221,240
k = 22	2.28624e+09	179,436,876	112,370,062	10,550,076
k = 23	2.31954e+09	181,982,115	114,505,744	10,825,761
k = 24	2.34792e+09	184,157,371	116,349,017	11,070,875
k = 25	2.37333e+09	186,108,431	117,999,580	11,294,439
k = 26	2.39664e+09	187,904,867	119,504,725	11,501,139
k = 27	2.41829e+09	189,580,382	120,891,039	11,693,180
k = 28	2.43849e+09	191,150,531	122,172,724	11,872,250
k = 29	2.45744e+09	192,629,345	123,363,397	12,039,734
k = 30	2.47529e+09	194,027,911	124,472,828	12,196,710

Table 8. A comparison of five *Bacillus* strains with the highest number of unique signatures and five others with the lowest number of signatures of length 18 within species and in the entire database. This table shows that within-species similarity and variability have more influence on the volume of signatures than the remainder of the database.

NAME OF STRAINS	WITHIN-SPECIES	IN THE ENTIRE DATABASE	THE ORIGINAL GENOME SIZE
Bacillus_megaterium_WSH_002_uid159841	4,860,315	4,012,591	5,0 MB
Bacillus_infantis_NRRL_B_14911_uid222804	4,712,042	3,932,760	4,8 MB
Bacillus_1NLA3E_uid81841	4,527,694	3,734,930	4,7 MB
Bacillus_cellulosilyticus_DSM_2522_uid43329	4,441,938	3,688,824	4,6 MB
Bacillus_clausii_KSM_K16_uid58237	4,177,156	3,576,848	4,2 MB
Bacillus_subtilis_168_uid57675	248	205	4,1 MB
Bacillus_amyloliquefaciens_CC178_uid226115	247	202	3,8 MB
Bacillus_anthracis_A0248_uid59385	0	0	5,4 MB
Bacillus_anthracis_A2012_uid54101	0	0	284 KB
Bacillus_anthracis_Ames_Ancestor_uid58083	0	0	5,4 MB

database, *Bacillus* species with 81 strains in the database was selected. The three phases of the pipeline were executed on these strains. Table 8 contains five strains of *Bacillus* with the highest number of unique signatures and five others with the lowest number within species and in the entire database. Although *Bacillus_anthracis_A0248_uid59385* and *Bacillus_anthracis_Ames_Ancestor_uid58083* have larger genome size, any unique signature of length 18 could not be found for them because of their high similarity with other *Bacillus* strains.

Results on both forward and reverse-complement sequences of bacterial genome. In the genome databases such as NCBI, only one strand of DNA sequence is provided. However, to design the primers, both forward and reverse-complement sequences should be considered. Moreover, depending on the sequencing technology, generated short reads can be from both strands. Therefore, the ability to obtain DNA signatures of both strands is potentially useful.

For the reverse-complement sequences, the size of the output files and the computational times of the first and second phases of the pipeline were the same as in the forward genome implementations. The output of the second phase for forward and reverse-complement genome databases resulted in a file of 103.03 GB for each. We have repeated WordCount on both of the databases one more time to determine the frequencies of *k*-mers as illustrated in Figure 4. The final result was a file of 52.53 GB for the forward and the same size for the reverse-complement genome database. On the one hand, it means that the volume of data containing unique signatures for the forward database decreased from 67.5 to 52.53 GB similarly to the reverse-complement genome database. On the other hand, the overall volume of DNA signatures that we could find for the species in the bacterial genome database increased from 67.5 GB containing signatures for a single strand to 105.06 GB for both strands of DNA.

Implementation results for the forward and reverse-complement bacterial genome database and the human genome database. We have considered the forward bacterial genome as the target database. We have applied the method that is described in Figure 4 and found 50.28 GB of *k*-mers for the target genome database which are unique among the three databases.

Table 9 presents the file size and numbers of unique DNA signatures of the target database against the nontarget ones.

Performance evaluation and computational times. For this experiment, we have applied two different platforms. The first one was a single node with 12 processors of Intel Core i7-4930K CPU at 3.40 GHz and 55 GB of RAM and 6 TB of hard disk. The operating system was Ubuntu 12.04.5 LTS, Java SE Version "1.8.0–25", Hadoop Version 1.2.1, and Hive-0.12.0. We have installed this node as a single-node Hadoop cluster. Another platform was a multi-node cluster with seven nodes including the master node and six slave nodes. The master node was an Intel Core2 Quad CPU Q6600 at 2.40 GHz and 8 GB of RAM and 3.2 TB of hard disk, while slaves had 4 GB of RAM, Intel Core i3-2100 CPU at 3.10 GHz and 500 GB of hard disk, all with the desktop version of Ubuntu 14.04.1 LTS 64-bit, Java Version "1.7.0–65" OpenJDK, Hadoop 12.1, and Hive-0.12.0.

The first phase of the pipeline executed with GkmerG took 156 minutes with five nodes and 780 minutes with a

Table 9. Number of unique DNA signatures for the forward bacterial genome database as the target and two other nontarget databases.

DATABASES	FILE SIZE	NUMBER OF SIGNATURES
Unique signatures of the Forward bacterial genome database	67.5 GB	3,552,866,254
Forward + Reverse-Complement bacterial genome databases	52.53 GB	2,764,759,739
Forward + Reverse-Complement bacterial genome + Human genome databases	50.28 GB	2,646,494,945

Table 10. A comparison of computational results of the first and second platforms in the second and third phases of the pipeline in order to find unique DNA signatures and their related species in the forward genome database (time in minutes).

STEPS	FILE SIZE (GB)	TIME FOR THE FIRST PLATFORM	TIME FOR THE SECOND PLATFORM
Copy *k*-mers generated by GkmerG to the HDFS	177.35	60	63
WordCount process	177.35	447	1169
Copy the result from HDFS to a local directory	103.03	34	27
Extracting unique signatures and creating tables in Hive	67.5	60	60
Loading unique signatures to the Hive table	67.5	23	26
Loading *k*-mers and reference numbers to the Hive table	220.35	79	83
Executing the queries and copy the result to a local directory	83.83	1120	959
Total computational time		1823	2387

single node from the second platform to generate 18-mers from the original bacterial genome database (9.7 GB). As an output, we got 2,773 files containing 18-mers with a total size of 177.35 GB.

Table 10 contains the corresponding computational results and the size of the files in the second and third phases of the pipeline for both platforms.

Although the whole computation on the second platform took about nine hours more than on the first one, comparing the RAM and CPU capacity of the two platforms confirms the ability of a cluster of low-cost computers that are commonly available in research facilities to accelerate big data analytics.

Table 11 compares the size of the files for frequencies 1–3 and the time of loading and processing the queries on the first platform. The file containing 220.35 GB data was used as the second table for all the implementations.

Conclusions

Data obtained in this study clearly show the efficiency of our proposed pipeline to find all possible DNA signatures of a target database. In this pipeline, we intend to overcome some limitations of DNA signature discovery by focusing on efficiency issues to detect all the possibilities of unique and common DNA signatures in a database, regardless of such challenges as pairwise alignment and mismatch tolerance. Another important feature of this pipeline is its ability to select target and nontarget databases. From the standpoint of this research, nontarget genome database is not necessarily defined as the entire background genome databases such as BLAST for the assessment and specificity evaluation of DNA signatures. It can be determined due to the requirements. General applicability is another issue that is considered in this pipeline; it can be launched either in a cluster of low-cost nodes or in a HPC environment. Although the volumes of the datasets in this study are very large (eg, 287.85 GB in a single run), DNA signatures are detected very precisely and comprehensively in the target databases and the execution times are reasonably short. The proposed experiment is just the basic idea, and there is a great flexibility to design implementations for phases of this approach. Once the pipeline is implemented, the users will find how to manipulate their datasets according to the requirements. This pipeline can be an efficient method, not only for DNA signature discovery but also for other purposes in bioinformatic and metagenomic studies such as the alignment and assembly of short reads and next-generation sequencing analysis.

Table 11. A comparison of loading and execution times of the frequencies 1–3 in the third phase.

FREQUENCY	1	2	3
Size of the file containing signatures (GB)	67.5	13.1	4.66
Time for loading file into the Hive table (minutes)	23	4	1
Execution time and copy the result to local directory (minutes)	1120	661	557

Acknowledgments

The authors would like to gratefully thank Samira Sajjadian, Edéné Rutkovszky, Herendi Tamás, and Laszló Kovács for their help, support, and encouragement. Special thanks also to the Faculty of Informatics and Department of Computer Graphics and Image Processing at the University of Debrecen for providing computational resources for this research.

Author Contributions

Conceived and designed the experiments: RK, AH. Analyzed the data: RK. Wrote the first draft of the manuscript: RK. Contributed to the writing of the manuscript: RK, AH. Agreed with the manuscript results and conclusions: RK, AH. Jointly developed the structure and arguments for the paper: RK, AH. Made critical revisions and approved final version: RK, AH. Both authors reviewed and approved the final manuscript.

Supplementary Materials

Download source for GkmerG and supplementary data:

1 http://www.inf.unideb.hu/~hajdua/HTSFinder.html
2 https://sourceforge.net/projects/htsfinder/
3 https://github.com/raminkm/HTSFinder

Content (after decompression):

- Hadoop and Hive installation guide and command lines.
- Excel file of bacterial genome database reference generated by GkmerG.
- List of bacterial genomes without any unique 18-mers (DNA signature) in the database.
- The GkmerG algorithm figure.
- GkmerG.tar.gz including software components and an example of database for testing.

REFERENCES

1. Kaderali L, Schliep A. Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics*. 2002;18(10):1340–9.
2. Francois P, Charbonnier Y, Jacquet J, et al. Rapid bacterial identification using evanescent-waveguide oligonucleotide microarray classification. *J Microbiol Methods*. 2006;65(3):390–403.
3. Li F, Stormo GD. Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics*. 2001;17(11):1067–76.
4. Coenye T, Vandamme P. Use of the genomic signature in bacterial classification and identification. *Syst Appl Microbiol*. 2004;27(2):175–85.
5. Větrovský T, Baldrian P. The variability of the 16S rRNA gene in bacterial genomes and its consequences for bacterial community analyses. *PLoS One*. 2013;8(2):e57923.
6. Wooley JC, Godzik A, Friedberg I. A primer on metagenomics. *PLoS Comput Biol*. 2010;6(2):e1000667.
7. Tembe W, Zavaljevski N, Bode E, et al. Oligonucleotide fingerprint identification for microarray-based pathogen diagnostic assays. *Bioinformatics*. 2007; 23(1):5–13.
8. Satya RV, Zavaljevski N, Kumar K, Reifman J. A high-throughput pipeline for designing microarray-based pathogen diagnostic assays. *BMC Bioinformatics*. 2008;9(1):185.
9. Satya RV, Kumar K, Zavaljevski N, Reifman, J. A high-throughput pipeline for the design of real-time PCR signatures. *BMC Bioinformatics*. 2010;11(1):340.
10. Vijaya Satya R, Zavaljevski N, Kumar K, et al. In silico microarray probe design for diagnosis of multiple pathogens. *BMC Genomics*. 2008;9(1):496.
11. Phillippy AM, Ayanbule K, Edwards NJ, Salzberg, S.L. Insignia: a DNA signature search web server for diagnostic assay development. *Nucleic Acids Res*. 2009;37(Web Server issue):W229–34.
12. *Insignia Database and Web Interface*. Available at: http://insignia.cbcb.umd.edu/.
13. Kurtz S, Phillippy A, Delcher AL, et al. Versatile and open software for comparing large genomes. *Genome Biol*. 2004;5(2):R12.
14. Bader KC, Grothoff C, Meier H. Comprehensive and relaxed search for oligonucleotide signatures in hierarchically clustered sequence datasets. *Bioinformatics*. 2011;27(11):1546–54.
15. Lee HP, Sheu T-F, Tang CY. A parallel and incremental algorithm for efficient unique signature discovery on DNA databases. *BMC Bioinformatics*. 2010;11(1):132.
16. Lee HP, Sheu TF, Tsai YT. Efficient discovery of unique signatures on whole-genome EST databases. In: Proceedings of the 2005 ACM Symposium on Applied Computing; New Mexico, USA. 2005:100–4.
17. Zheng J, Close TJ, Jiang T, Lonardi, S. Efficient selection of unique and popular oligos for large EST databases. *Bioinformatics*. 2004;20(13):2101–12.
18. Lee HP, Huang Y-H, Sheu T-F. Rapid DNA signature discovery using a novel parallel algorithm. In: ICCGI 2012, The Seventh International Multi-Conference on Computing in the Global Information Technology; Venice, Italy. 2012:83–8.
19. Lee HP, Sheu T-F. An algorithm of discovering signatures from dna databases on a computer cluster. *BMC Bioinformatics*. 2014;15(1):339.
20. Marcais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*. 2011;27(6):764–70.
21. Kaderali L, Schliep A. An algorithm to select target specific probes for DNA chips. *Bioinformatics*. 2002;18(10):1340–9.
22. Rouillard JM, Zuker M, Gulari E. OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach. *Nucleic Acids Res*. 2003;31(12):3057–62.
23. Wernersson R, Nielsen HB. OligoWiz 2.0 – integrating sequence feature annotation into the design of microarray probes. *Nucleic Acids Res*. 2005;33(suppl 2): W611–5.
24. Nordberg EK. YODA: selecting signature oligonucleotides. *Bioinformatics*. 2005;21(8):1365–70.
25. Ashelford KE, Weightman AJ, Fry JC. PRIMROSE: a computer program for generating and estimating the phylogenetic range of 16S rRNA oligonucleotide probes and primers in conjunction with the RDP – II database. *Nucleic Acids Res*. 2002;30(15):3481–9.
26. Ludwig W, Strunk O, Westram R, et al. ARB: a software environment for sequence data. *Nucleic Acids Res*. 2004;32(4):1363–71.
27. Adams MD, Kelley JM, Gocayne JD, et al. Complementary DNA sequencing: expressed sequence tags and human genome project. *Science*. 1991;252(5013): 1651–6.
28. Baxevanis AD, Ouellette BF. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*. Vol. 43. John Wiley & Sons; 2004.
29. Choudhary M, Mackenzie C, Nereng KS, Sodergren, E., Weinstock, G.M., Kaplan, S. Multiple chromosomes in bacteria: structure and function of chromosome II of Rhodobacter sphaeroides 2.4. 1t. *J Bacteriol*. 1994;176(24):7694–702.
30. *Apache Hadoop*. Available at: http://hadoop.apache.org/.
31. White T. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc.; 2012.
32. Cloudera. *Hadoop and Big Data*. Available at: http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html.
33. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium On; Incline Village, Nevada, USA. 2010:1–10.
34. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. *Commun ACM*. 2008;51(1):107–13.
35. Battre D, Ewen S, Hueske F, Kao O, Markl V, Warneke D. Nephele/pacts: a programming model and execution framework for web-scale analytical processing. In: Proceedings of the 1st ACM Symposium on Cloud Computing; Indianapolis, IN, USA. 2010:119–30
36. *Apache Hadoop NextGen MapReduce (YARN)*. Available at: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.
37. Capriolo E, Wampler D, Rutherglen J. *Programming Hive*. O'Reilly Media, Inc.; 2012.
38. Thusoo A, Sarma JS, Jain N, et al. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*. 2009;2(2):1626–9.
39. *Apache Hive*. Available at: https://hive.apache.org/.
40. Ulrich RL, Ulrich MP, Schell MA, Kim, H.S., DeShazer, D. Development of a polymerase chain reaction assay for the specific identification of *Burkholderia mallei* and differentiation from *Burkholderia pseudomallei* and other closely related Burkholderiaceae. *Diagn Microbiol Infect Dis*. 2006;55(1):37–45.
41. Godoy D, Randle G, Simpson AJ, et al. Multilocus sequence typing and evolutionary relationships among the causative agents of melioidosis and glanders, *Burkholderia pseudomallei* and *Burkholderia mallei*. *J Clin Microbiol*. 2003;41(5):2068–79.