

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER THESIS No. 1123

**A reduced gene database for
precision species detection**

Dorija Humski

Zagreb, June 2015.

Zagreb, 6. ožujka 2015.

DIPLOMSKI ZADATAK br. 1123

Pristupnik: **Dorija Humski (0036455326)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Reducirana baza gena za precizno određivanje vrsta**

Opis zadatka:

Korištenje alata za pretraživanje bioloških baza podataka je čest pristup za određivanje vrsta iz meteganomskih i metatranskriptomskih uzoraka. Međutim, zbog sličnosti između vrsta i dijeljenog genetskoga materijala u bazi i sekvenciranom uzorku taj pristup je često neodgovarajući. Glavni cilj ovoga rada je poboljšanje preciznosti određivanja vrsta umanjujući sličnost između zapisa u bazi gena. Ovaj pristup treba:

- 1) Grupirati gene koristeći metode temeljene na poravnanju i metode koje ne koriste poravnanje;
- 2) Konstruirati novu bazu podataka koristeći predstavnike grupa;
- 3) Implementirati modul za precizno određivanje sojeva iz egzaktnoga poravnanja na bazu.

Grupiranje je potrebno provesti na bazi gena mikroba. Grupe moraju biti tako kreirane da je svaki gen unutar grupe maksimalno p% različit od predstavnika grupe. Geni koji ne predstavljaju grupu trebaju biti predstavljeni razlikom u odnosu na predstavnika. Modul za precizno određivanje sojeva će koristiti te razlike u cilju određivanja najvjerojatnijeg soja u uzorku analizirajući poravnanje očitavanje iz uzorka s reduciranom bazom. Rješenje mora biti prilagođeno paralelnoj arhitekturi i napisano u jeziku C++ . Programski kod je potrebno komentirati i pri pisanju pratiti stil opisan u Googleovom C++ vodiču. Kompletnu aplikaciju postaviti na Github pod jednom od OSI odobrenih licenci.

Zadatak uručen pristupniku: 13. ožujka 2015.

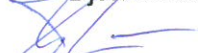
Rok za predaju rada: 30. lipnja 2015.

Mentor:



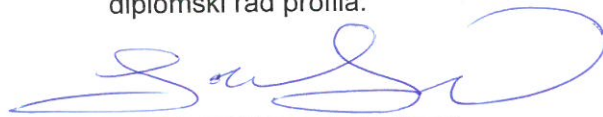
Izv. prof. dr. sc. Mile Šikić

Djelovođa:



Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za
diplomski rad profila:



Prof. dr. sc. Siniša Srblijić

Zagreb, 6 March 2015

MASTER THESIS ASSIGNMENT No. 1123

Student: **Dorija Humski (0036455326)**
Study: Computing
Profile: Computer Science

Title: **A reduced gene database for precision species detection**

Description:

It is a common approach to use database search tools to infer species from metagenomic and metatranscriptomic samples. However, given the species similarity and shared genetic content present in both the database and the sequenced sample, this approach displays an obvious fault. The main purpose of this work will be to improve upon the precision of species detection reducing the similarity among the entries in the gene database. The approach will:

- 1) Perform gene clustering using alignment and alignment-free similarity measures;
- 2) Construct a novel database from cluster representatives;
- 3) Implement a module for precise strain identification from exact alignment against the database.

The clustering is to be performed on a microbial gene database. The clusters should be created so that each gene within a single cluster is at most $p\%$ different from the representative gene. The non-representative genes within a cluster will be represented as a difference from one representative gene. The precise strain identification module will use these differences to deduce the most probable strains present in the sample by analyzing the alignment of sample reads to the reduced database. The solution should be appropriated for parallel architectures and implemented in C++. The code is to be documented using comments and should follow the Google C++ Style Guide when possible. The complete application should be hosted on Github under an OSI approved licence.

Issue date: 13 March 2015
Submission date: 30 June 2015

Mentor:



Associate Professor Mile Šikić, PhD

Committee Chair:



Full Professor Siniša Srbljić, PhD

Committee Secretary:



Assistant Professor Tomislav Hrkać, PhD

Hvala mom mentoru na podršci i strpljenju zadnjih par godina.

*I would like to thank Professor Bernard Moret for the help, advices and support.
But the most for accepting me in the LCBB family.*

Hvala mojoj obitelji na beskrajnoj podršci i pomoći svih ovih godina.

CONTENTS

1. Introduction	1
2. Materials and Methods	3
2.1. Algorithms for Clustering data	3
2.1.1. Hierarchical clustering	4
2.1.2. Partitional clustering	5
2.2. Progressive Multiple alignment	6
2.3. HMMs profiles	7
3. Implementation	9
3.1. Single-link clustering	9
3.2. Hierarchical clustering and multiple alignment	12
3.3. Clustering using UCLUST	15
3.4. Final design	17
3.4.1. Example	19
4. Results	20
5. Conclusion	24
Bibliography	26

1. Introduction

Genome sequencing is becoming cheaper, faster and better, and as a result a huge amount of sequences are already produced. Whilst having a huge amount of data improves accuracy of data analysis, the problem comes, however, when there is a need to produce something quickly.

From the observations of the databases, it has been noticed that some sequences in the database are similar to each other. The similarity between sequences, in the sense of alignments against other sequences, often implies a redundancy. While the redundancy implies increasing in a computational cost and decreasing a speed of searching a sequence in the database. The idea of excluding redundant sequences has been born, but the problem which has left was "How?".

How to keep accuracy at the same level, but increase speed of the operations using database? The answer on the question was to create a reduced database. The reduced database is modeled in a way that at first it gives us information only about significant sequences, and all the other sequences are hidden behind their representative. The way of creating a database like this, is a process of a clustering.

To produce a reduced database for the purpose of the faster identification of the species, we have applied a couple of approaches. The main idea was to model a database as a hierarchical clusters, hence we have applied two different approaches of hierarchical clustering. The issues we came to were how to make a good clusters and what should be a representative of the cluster. The third approach involves partitional clustering, where the sequences from the database are divided into partitions according to the similarity. The issues for partitional clustering were the same as the ones for hierarchical clustering. Since each approach implies different model of the database, we associate to them different modules for an identification of the species.

The thesis is organized as follows: Chapter 2 describes methods and algorithms important for understanding the other parts of the thesis. Chapter 3 is divided into four sections, the first tree sections describes different approaches while trying to solve a problem, while the four section contains the details of the final design. Chapter 4

consists of results obtained while testing our approaches and Chapter 5 highlights the conclusions of our work.

2. Materials and Methods

Since Implementation chapter describes algorithms based on clustering methods, progressive multiple alignment and HMM profiles, understanding of these topics is relevant for understanding the rest of the thesis. Hence this chapter contains a short description of the topics.

2.1. Algorithms for Clustering data

Clustering is the process of classifying objects into subsets that have a meaning in the context of a particular problem (Anil K. Jain, 1988). Classifying should be done in such a way that objects in one cluster are more similar to each other than to objects in other clusters. The goal of the clustering is to reduce the amount of data by grouping similar data together. The clustering is imposed on a finite set of objects, where the relationship between objects is known. The relationship between objects should be a measurable value which represents a distance or proximity. A matrix representing such a relationship is typically called *Proximity matrix*. Besides objects, the proximity matrix is the only input to a clustering algorithm.

Algorithms for clustering can be *divisive* or *agglomerative*. Divisive algorithm starts with all objects in one cluster and continues with subdividing existing clusters into smaller pieces according to the proximity matrix. Agglomerative algorithm reverses the process by starting with each object divided into its own cluster and continues with merging existing clusters into larger pieces.

According to the type of structure imposed on the data, clustering methods are divided into two categories:

- Hierarchical clustering
- Partitional clustering

The hierarchical clustering is a nested sequence of partitions, whereas the partitional clustering is a single partition.

2.1.1. Hierarchical clustering

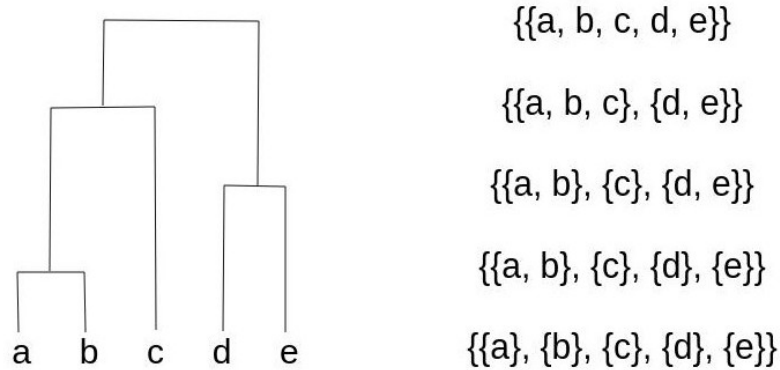


Figure 2.1: Example of hierarchical clustering

The hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence. An example of hierarchical clustering is shown in figure 2.1. A tree structure used for the representation is a *dendrogram*, and each layer in the dendrogram represents one cluster. The dendrogram enables us to see how objects are being merged into clusters or split at successive levels of proximity. Two specific hierarchical clustering methods are called *single-link* and *complete-link* methods (Anil K. Jain, 1988). The difference between the single-link and the complete-link can be seen on the problem: suppose that objects $\{x_1, x_2, \dots, x_n\}$ are divided into clusters $\{C_1, C_2, C_3, \dots, C_m\}$ and that the next step in clustering is to merge clusters C_i and C_j . The relationship between objects is represented as a distance, $d(x, y)$. Depending on the method which is used for clustering, one of the two statements is true:

- Single-link: $\min_{x \in C_i, y \in C_j} \{d(x, y)\} = \min_{r \neq s} \{ \min_{x \in C_r, y \in C_s} \{d(x, y)\} \}$
- Complete-link: $\max_{x \in C_i, y \in C_j} \{d(x, y)\} = \min_{r \neq s} \{ \max_{x \in C_r, y \in C_s} \{d(x, y)\} \}$

Where $r, s \in 1, 2, \dots, m$.

According to those statements, the single-link method is also known as "minimum" method, while the complete-link method is known as "maximum" method.

One possible algorithm for the single-link method is "Graph theory algorithm for single-link clustering" (Anil K. Jain, 1988). The algorithm is given in Algorithm 1. The relationship between objects is represented by distance, thus the algorithm use a dissimilarity matrix as an input. It is the agglomerative algorithm based on the minimum spanning tree (MST).

Data: Objects, dissimilarity matrix

1. Place each object in its own cluster
2. Create a weighted graph where nodes are clusters. Weight is dissimilarity between two nodes (clusters)
3. Find MST on the weighted graph
4. Merge two clusters connected by the MST edge with smallest the weight
5. Replace the weight of the edge selected in step 4 by a weight larger than the largest dissimilarity

Repeat steps 4. and 5. until all objects are in one cluster

Algorithm 1: Graph theory algorithm for single-link clustering

2.1.2. Partitional clustering

The main goal of partitional clustering is to partition the objects into K clusters in such a way that objects in a cluster are more similar to each other than to objects in different clusters. A value of K may or may not be specified (Anil K. Jain, 1988). Clustering is done according to a defined clustering criterion. The criterion can be *local* or *global*. The local criterion uses information of a local structure of the data and utilizes that information in a clustering process. The global criterion uses information from prototypes which represent each cluster, classifying objects according to the most similar prototypes. The criterion is highly related to a problem and must be simple enough due to the computational cost but yet complex enough to reflect data structures.

Many clustering methods are designed for a variety of problems, including clustering of nucleotides or protein sequences, and one of the possible algorithms is UCLUST, described below.

UCLUST

UCLUST is a greedy algorithm designed for clustering nucleotide or protein sequences (RC, 2010). The algorithm is partitional, using a global criterion and does not have a specified number of clusters. A prototype of the cluster is a sequence from a dataset, which is called a centroid. For a given threshold, T , UCLUST is designed to find a set of clusters such that:

- All centroids have similarity $< T$ to each other, and
- All member sequences have similarity $\geq T$ to a centroid.

Similarities between sequences are defined as an identity between those sequences. To calculate the identity, UCLUST uses global alignment.

The algorithm can be stated as follows. Given n sequences, process the sequences one by one in the order they appear. Match each sequence to all existing centroids. If an identity between a sequence and some of the centroids is higher than a threshold value T , the sequence is assigned to that cluster, otherwise the sequence becomes the centroid of a new cluster.

Since the sequences are processed one by one in the order they appear, the order plays crucial role in defining a cluster. This means that sequences should be ordered in such a way that the most appropriate centroids tend to appear earlier in the file (RC, 2010). One way to do this is to sort sequences by length, although the best order depends on the application.

2.2. Progressive Multiple alignment

Multiple sequence alignment (MSA) is the central technique for inferring biological information from a set of sequences. The MSA involves an alignment of more than two sequences and aims to find equivalent positions across an aligned query set of the sequences (V. et al., 2003). The MSA can provide a wealth of information about structure-function relationships within a set of the sequences.

MSA can be made using Progressive algorithms. Progressive algorithms are based on three basic steps:

1. Calculating a matrix of pairwise distances based on pairwise alignments between the sequences
2. Use the pairwise distance matrix and build a "guide tree"
3. Use the "guide tree" to guide multiple alignment

Progressive algorithms build the multiple alignment by the iteration of pairwise alignment in the internal nodes of a predefined guide tree (A. and N., 205). The alignment progresses from the terminal nodes toward the root of tree. The order of alignment is derived from pairwise alignment scores and a calculation of the "guide tree". The calculation of the "guide tree" is a process of hierarchical clustering of the sequences based on their pairwise alignment score. The result of the hierarchical clustering is a dendrogram, referred to as the "guide tree". The resulting branch order of the "guide tree" is then followed to align the sequences, such that the most similar sequences are aligned first, and gradually the more distant sequences are included in the growing MSA (V. et al., 2003).

Various programs are based upon progressive alignment strategy. They vary according to how they calculate pairwise alignment, building the "guide tree" and iterating through the "guide tree" to produce multiple alignments.

2.3. HMMs profiles

A profile is a description of the consensus of the multiple sequence alignment. Profile Hidden Markov Models (HMMs) turn a multiple sequence alignment into a position-specific scoring system suitable for searching databases (Eddy, 1998). Profile HMMs have several advantages over standard profiles. While standard profiles use the raw frequency observed from the data to assign the score for the residue, HMMs apply a statistical method. Beside the fact of having a formal probabilistic basis, HMMs profiles have a consistent theory behind a gap and insertion scores, in contrast to standard profile methods which use heuristic methods (Profile HMM Analysis).

To build HMMs profiles, for each consensus column of multiple alignment, three states should be created: "match", "insert" and "delete" state. A "match" state models the distribution of allowed residues in the column. An "insert" state allows insertion of one or more residues between the column and the consecutive column, while "delete" state is used for deleting the consensus residue.

Two types of probabilities are associated with HMMs profiles: *transition* probability and *emission* probability. Transition probabilities describe transition from one state to another, while emission probabilities are associated with a particular state (particular "match" state).

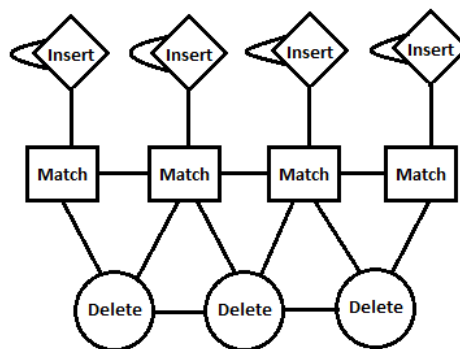


Figure 2.2: HMM profiles

To give a better insight, a simplified HMM profile can be seen on the figure 2.2. A consensus of the MSA consists of 4 columns, and each of them is represented by three states.

Aligning of a new sequence to a profile HMM is done by finding the most probable path that the sequence may take through the model, using the transition and emissions probabilities to score each possible path.

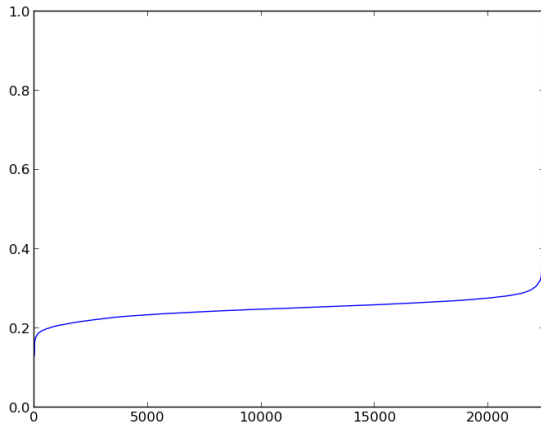
3. Implementation

For the purpose of producing a reduced database, we have applied three different approaches. The first two are based on a hierarchical clustering, while the third one is a partitional clustering solution. Approaches are named according to the methods and algorithms they use, thus there are: Single-link clustering, Hierarchical clustering using multiple alignment and Clustering using UCLUST. This chapter is designed to provide an overview of the implemented methods. It also gives advantages and disadvantages of the methods and the reason why some of the methods are not appropriate.

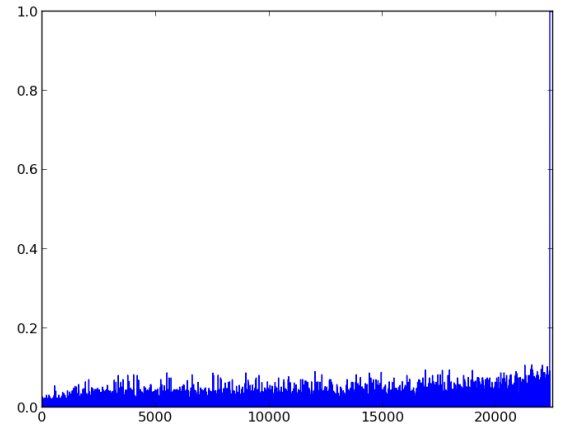
3.1. Single-link clustering

The first approach in solving a problem is to incorporate a pairwise alignment and a single-link clustering. The main idea behind the approach is to determine a proximity matrix using the pairwise alignment and create a dendrogram with the single-link clustering algorithm. Each node in the dendrogram is represented by a profile of the sequences which belong to that node. To identify a new sequence, the dendrogram is searched from the root to leaves. Since each node in the dendrogram is represented by a profile, a score between the node and the new sequence is a matching score between the profile and the sequence. Searching could be optimized by using a greedy algorithm. The algorithm would test a matching score from the left and right branch of the current node, and choose the side where the score is higher, but only if the difference between those two scores is large enough. If the difference is not significantly large enough, the algorithm continues searching the both sides.

To determine the proximity matrix, two methods were tested: Smith-Waterman and BLAST (Bet, 2008). Emboss-water is used as an implementation of Smith-Waterman algorithm (EMBOSS Smith-Waterman). The main goal was to prove that the BLAST score is sufficient enough for clustering and that there is no need to use the slower Smith-Waterman (SW) method. To be able to compare SW score and BLAST score, scores must be normalized and scaled to the range $[0, 1]$. BLAST bit-score is already



(a) Smith-Waterman score



(b) BLAST score

Figure 3.1: Pairwise alignment scores

normalized, but not scaled. Scaling is done by using the expression:

$$scaled_bit_score(Q, S) = \frac{bit_score(Q, S)}{\max_{q \in database} \{bit_score(q, S)\}}$$

Since the SW score is not normalized nor scaled, normalization is done by dividing the score with the length of alignment and scaling by dividing with the "matching" value. The expression is:

$$SW_score(Q, S) = \frac{SW_score(Q, S)}{alignment_length * MATCHING_VALUE}$$

The testing was made on different datasets and the results varied, however general conclusion can be made. The figure 3.1 shows the results for 150 sequences, where they were aligned to each other with BLAST and SW. On the horizontal axis are shown pairs of the sequences (an index associated with the pair), while on the vertical axis are shown corresponding similarity scores. The similarity scores were normalized and scaled. While in the figure 3.1a there is an increasing trend, in the figure 3.1b function shows growth even though it is not continuous. The reason is that BLAST would marked an alignment between some sequences as not significant, according to the E-value. The conclusion was that the BLAST score is sufficient enough for clustering.

Once the proximity matrix has been made, the clustering algorithm can be applied to create a dendrogram. For clustering the algorithm *Graph theory algorithm for single-link clustering* is used. The algorithm is described in Chapter 2.1.1. Since the algorithm is suitable for dissimilarities, the proximity matrix is converted into a dissimilarity matrix by replacing each value a_{ij} into $a_{ij} = 1 - a_{ij}$. A clustering process

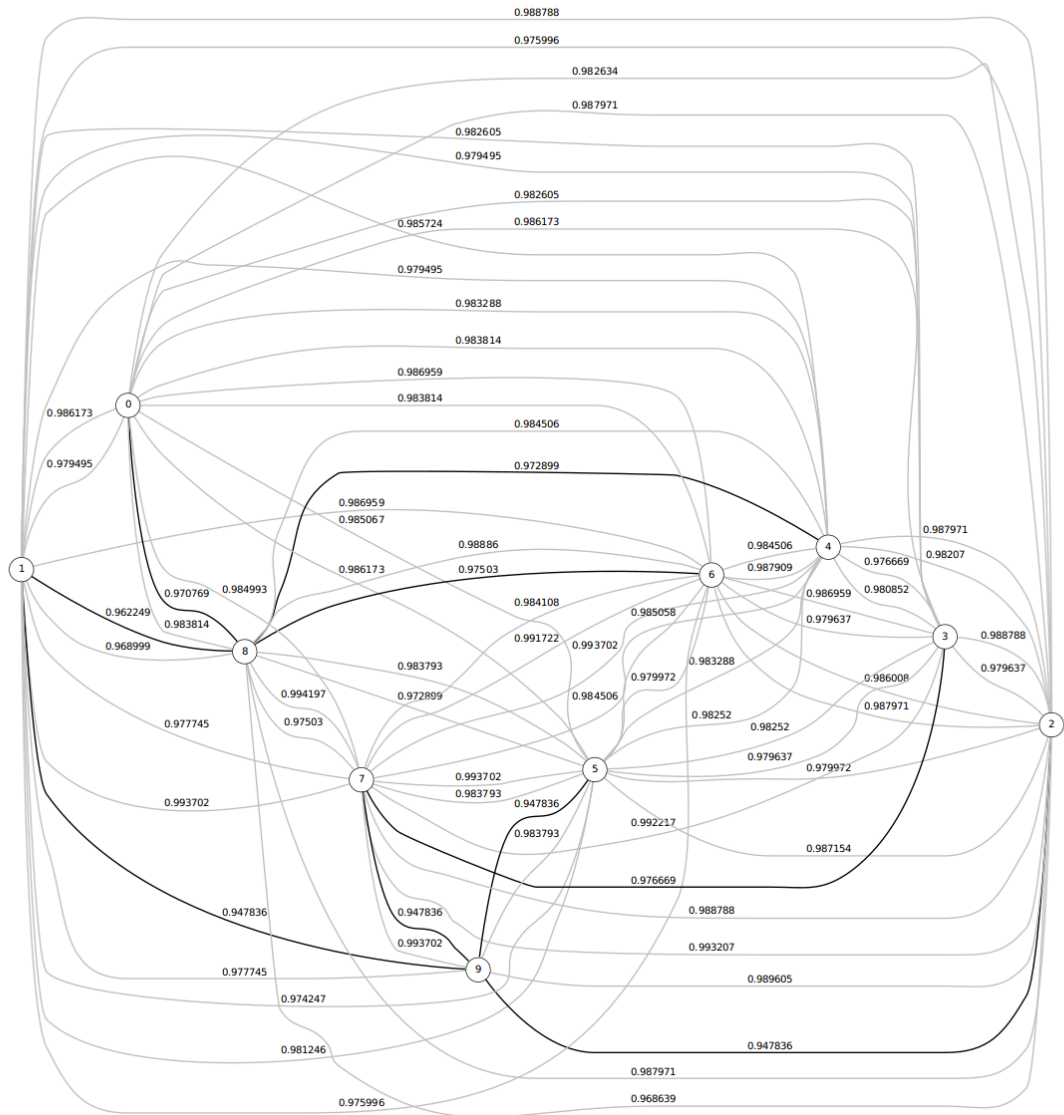


Figure 3.2: Weighted graph and the MST

for the small dataset of 10 sequences is shown in figures 3.2 and 3.3. The weighted graph is illustrated in the figure 3.2, a node represents the sequence, and an edge represents dissimilarity between sequences. Bold edges are part of the MST. An associated dendrogram is shown in the figure 3.3.

Running on the different datasets, with different dissimilarity matrices, a dendrogram tends to be very unbalanced. An unbalanced dendrogram means that each level left/right branch would be represented by a profile made from only one sequence. While searching for a new sequence, we still have to go through all the sequences from the dataset. Since our goal is to reduce the database, in a way that while searching we do not have to go through all the sequences from the database. Therefore, we

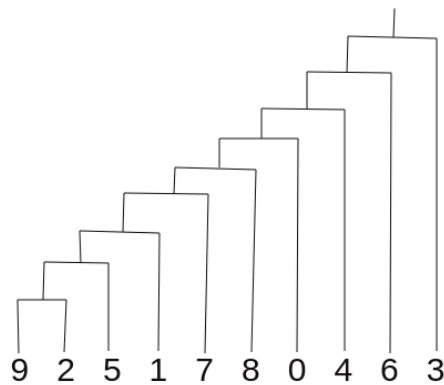


Figure 3.3: Dendrogram

concluded that the single-link approach is not suitable for clustering sequences.

3.2. Hierarchical clustering and multiple alignment

The first approach was not appropriate for our problem, so we decided to move on in a different direction. Trying to involve State-of-the-Art algorithms as much as possible sounds promising. Since progressive multiple alignment algorithms use clusters of the sequences while producing an alignment, using these types of algorithms presents a possible solution to our problem.

The idea can be stated as follows:

1. Produce multiple alignment on the sequences.
2. Get the guide tree and all internal alignments.
3. From the internal alignments build HMM profiles.
4. Create a tree in which nodes are represented by HMM profiles, as a base using the guide tree.

For the purpose of an alignment we use Clustal Omega (F. et al., 2011). Clustal Omega is chosen since it is suitable for medium-large alignments and provides many options. By default it creates a distance matrix and a guide tree, but as an input it also accepts them. An option to accept the distance matrix is convenient in the case when we want to control a pairwise alignment. If we can find a better way to create a guide tree, but prefer that Clustal Omega makes alignments for us, there is an option to read the given guide tree.

HMM profiles are created using a package HMMER (Eddy and Wheeler, 2015). HMMER is used for a biological sequence analysis using profile hidden Markov models. We use options *hmmbuild*, *hmmcompress* and *hmmsearch*. *Hmmbuild* constructs profile HMMs from multiple sequence alignment(s). *Hmmcompress* prepares HMMs for *hmmsearch*, while *hmmsearch* searches profiles against a sequence database.

Data: Guide tree, profiles, parameter T , a new sequence

Result: List of potential sequences

Read the guide tree - guide tree contains indexes to the real HMM profiles;

$node = \text{Root}$;

Procedure {

if $node$ is a leaf **then**

 put in the List of potential sequences

else

 Read profile of the left child of the $node$;

$score_{left} = \text{score of matching the sequence to that profile}$;

 Read profile of the right child of the $node$;

$score_{right} = \text{score of matching the sequence to that profile}$;

if $(\text{abs}(score_{left} - score_{right}) > T)$ **then**

if $(score_{left} > score_{right})$ **then**

$node = \text{left}$;

else

$node = \text{right}$;

end

 go to the beginning of the *procedure*;

else

$node = \text{left}$;

 go to the beginning of the *procedure*;

$node = \text{right}$;

 go to the beginning of the *procedure*;

end

end

}

Algorithm 2: Searching for a sequence in the database

A new database contains HMM profiles for each node in the guide tree and guide tree with indexes to the HMM profiles. The procedure for finding a new sequence in the new database is presented in Algorithm 2. Parameter T states the threshold while choosing a side. For a smaller T , the procedure is more accurate since it goes through

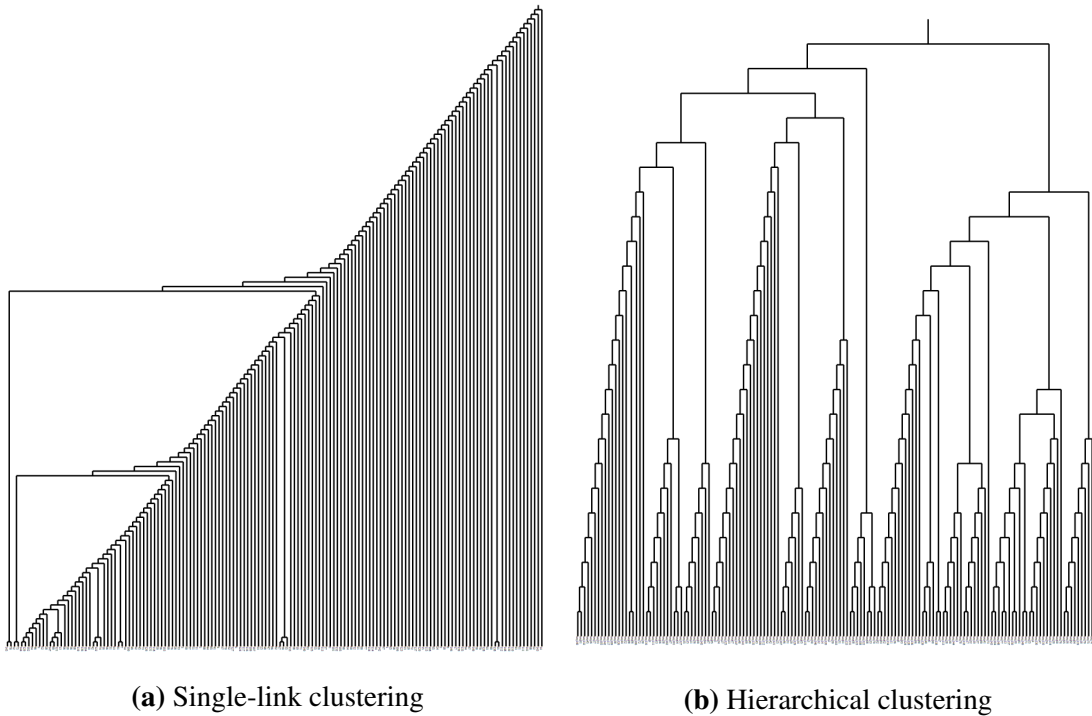


Figure 3.4: Dendrograms

a larger part of the tree, but the cost is time.

To prove that the second approach gives a better results than the first one, we can compare their dendrograms. The two dendrograms, shown on the figure 3.4, are built using single-link clustering (3.4a) and hierarchical clustering (3.4b). Algorithms were run on the dataset of 150 sequences. The unbalanced dendrogram in figure 3.4a increases the searching time of a new sequence.

The approach were tested on a set of bacterial coding sequences (CDS). In the table 3.1 are shown results. Since the method is exhaustive for a bigger dataset, we ran it on the dataset of 150 sequences and the dataset of 1500 sequences. Parameter T was equal to 100. The method shows high accuracy. We ran it on 10 different samples of datasets sizes 150 sequences and 1500 sequences, and the results were the same.

Table 3.1: Hierarchical clustering and multiple alignment

Dataset size	Accuracy	Time(s)
150	100%	0.14026
1500	100%	138.0

With the limitations of multiple alignment algorithms comes limitations of our approach. Some algorithms have a limited size of accepted datasets. If there is the

possibility to work with larger datasets, the problem then is accuracy. A solution for this is preprocessing, where the sequences first would be clustered by using some of the partitional clustering algorithms.

3.3. Clustering using UCLUST

Since the main part of our job is to reduce a dataset, our focus was to find the most appropriate solution to exclude some of the sequences and to keep only the most "significant" ones. The final and most appropriate solution is to incorporate some of the partitional clustering algorithms. The partitional clustering divides a larger dataset into smaller partitions represented by one of the sequences. The sequences which are representation of the partition (cluster) can be kept as the most "significant", and these sequences form a reduced database. There are many different algorithms to do clustering and each of them with different features. One of possible algorithms is UCLUST, see section 2.1.2. Since the features of UCLUST are suitable for our needs, it is a part of our solution.

The limitation of the non-commercial UCLUST version is that a maximum of 4Gb RAM can be used. Dataset of size 1Gb can use 4Gb RAM. Thus a bigger dataset is first divided into partitions of the maximum 50000 sequences, so UCLUST can be run on each partition. The set of 50000 sequences can also use more than 4Gb RAM, to prevent that, we added an extra constraint: if the size of a dataset reach the maximum of 1Gb, we divide the set into even smaller partitions. Since the clustering of one partition does not depend on the other partitions, the process of clustering the data in each partition can be run in parallel. At the end, the results of each partition are concatenated together. Since we concatenate centroids derived from different partitions, there is possibility that following criterion can be compromised: all centroids have similarity less than T to each other. To preserve that criterion, partitions are made in such a way to decrease the probability that two centroids coming from different partitions have a similarity more than T to each other. We can make this by dividing the database into partitions according to the length of the sequences. The sequences from the database are sorted by the length and then divided into partitions.

A size of the clusters mainly depends on the threshold (T). Whether an identity between a next sequence and existing centroids is smaller or bigger than the threshold, the sequence is classified as a new centroid or as a part of the cluster. For the smaller threshold, clusters are usually bigger, but the accuracy of such a cluster is questionable.

During a clustering process, UCLUST does not update centroids. Once a sequence

Table 3.2: Centroids

Total number	Real centroids	Percentage
48023	47375	98.65%
32082	31193	97.22%
38112	37391	98.10%
35937	35836	99.71%

```
>seq24
ATGAGCGTCAACGTCATCTTCGGCTCCGACGGTGGTAACACCAAGGCGGTCGCCTCCCGCATCGCCAAACACATCCAGGG
TCGCGCGATCGACATCAAGGCGGGAACGTCGCGGACTTCGAAAGCCCAGCCTGCTGATACTCGGATCGCCGACCTACG
GGCCCGCGATCTGCAGACCGATTGGGAGGCGCATCTCGACAAGCTGACCCAGGCCAAGCTCGCCGGAAGAAGGTGGCG
CTGTTCCGGCACCGGACAGGTCAGCTACCCGGACTGCTTCGTCGACCCATGGGCGTGCTCTATGATTTGGTCGTCTGA
GCAGGGCGTACCCTGGTCCGGCTTACCAGAGACCGCGGCTACGACTTACCAGGCTCAAGGCGGAGCGCGACGGCCAGT
>seq21
ATGACAAACATCGAACAGGATCGCTGGTCACGGGTTAAGGGCCGCTTGCCTCAAGCGTCGGTGAAGACGTTTACTCGAG
CTGGTTCGCCCCGATGGATCTGAAAGCGTGCAGCCCGAAAGCGTGACCTGTCCGTCGCCGACGCGATTCTGAAGAGCT
GGATCCAGACGCATTATCCGATCCGCTGCTGAGCTGCTGGCAGGCCGAGATGCCGAGGTGCATCGCGTCGACCTCACC
CCGACGAAGACAGGTCCGGAAGGCCCGCGCGGATCCGGTGATGTTCAATCCGACAGCCGCGTGGC
```

Figure 3.5: Centroids file

becomes the centroid of a cluster, it stays as the centroid (a representation of the cluster) even though it does not need to be a real centroid of the cluster. *Areal centroid* is a sequence from a cluster which has a minimum sum of distances to other sequences. The question was should we calculate a real centroid of the cluster and keep it as a representation of the cluster or are UCLUST centroids sufficient enough? We ran a test on different datasets and the results are shown in table 3.2. Since the percentage of the real centroids among UCLUST centroids is large enough, we decided to keep UCLUST centroids as a representation of the clusters.

An output of the clustering using UCLUST consists of two files: the file which contains centroids from clustering in FASTA format and the file with clusters. Clusters in the file are separated with a keyword: "Cluster". A name of the cluster is a name of the centroid and comes after the keyword. Each cluster is following with an empty row. Sequences which belong to the cluster are written in FASTA format after the name. An example of the output is shown in figures 3.5 and 3.6. Six sequences are clustered into two clusters, centroids re shown in the figure 3.5, while associated clusters are in figure 3.6.

Having centroids in a separate file makes the process of searching for a new sequence faster. When a new sequence has to be identified, the procedure can be stated as follows:

1. Make an alignment of the sequence to the centroids

```

Cluster:          seq24
>seq84
ATGCCGATACTCGTGGTCGGCGCCGGAGCGATCGCGGCTATTTCCGGCGGACGGCTGCTGCAGGCCGGCGCGATGTGAC
GTTCTTGGTGGCGCGAAACCGCCGCCGAACCTGCAACCGCACGGCCTGGTGATCAACAGCCCGCACGGGACGTACGC
TGGAGACCCCGCGGTCTCTGGCGACGGAGTTGAATCAGCCGTTTCGATCTGATCTGCTGAGCTGCAAGGCATTTCGAT
CTCGACGATGCCATCGCTTCGTTCCGGCCGGCGCCATGGCGTGTCTATGATTTGGTCTGTCGAGCAGGGCGCTACC
GTGGTCGGCTTACCAGACCCGGGCT
>seq66
ATGGCGATGCTTCCGCCGGATCAGATCAAGGGCGCCTTGGCGTTTCTGGCGAGGGCGGAGCGCTGAAGGACGTGCTGGC
CAGCGGCCACACCTCGAACGGCCGGCGAAAGCACCGCGAACACACCTGGCGGCTGTGCCTGATGGCGGTGCTGTTCCG
CCGATGATCTGGGCGATATCGACATCGCAAGCTTCTGAAAATCTGCATCGTGCACGACCTCGCGGAAGCGCTGCACGGC
GACATCCCGCGGATGATCAATCCGCCGACGGCAACAAGGGCGCGCAGGAGCGCGCCGATCTGGAAGCCCT
>seq24
ATGAGCGTCAACGTATCTTCGGCTCCGACGGTGGTAACACCAAGGGCGTGCCTCCCGCATCGCCAAACACATCCAGGG
TCGGCGGATCGACATCAAGGGCGGGAACGTGCGCGACTTCGAAAGCCCCAGCCTGCTGATACTCGGATCGCCGACCTACG
GCGCCGGCGATCTGCAGACCGATTGGGAGGCGCATCTCGACAAGCTGACCCAGGCAAGCTCGCCGGCAAGAAGTGGCG
CTGTTCCGCACCGCGACAGGTGAGCTACCCGGACTGCTTCGTCGACGCCATGGCGTGTCTATGATTTGGTCTGCGA
GCAGGGCGCTACCGTGGTTCACCGAGACCGGGCTACGACTTACCGGCTCCAAGGGCGAGCGCGACGGCCAGT
Cluster:          seq21
>seq21
ATGACAACATCGAACAGGATCGTGGTCACGGGTTAAGGGCCGCTTGGCGTCAAGCGTCGGTGAAGACGTTTACTCGAG
CTGGTTCCGCCGGATGGATCTGGAAAGCGTGCAGCCCGAAAGCGTGCACCTGTGGTCCCGACCGGATTCTGAAAGAGCT
GGATCCAGACCGATTATTCGATCGCTGCTGAGTGTGTCGAGGCCGAGATGCCGCGAGGTGCATCGCGTGCACCTCACC
CCGCAGCAAGACAGGTCCGGGAAGGCCCGCCGGCGATCCGGTGTGTTCAATCCCGACACCGCGCTGGC
>seq97
ATGACCGTCTCAAAGATCCATCGCTGACGCTGACGCATTTCCGCAATTATCGCGCGGCGAGCGTGACGGTGGCGGGCGAC
GTCGTGGTGGTGGTGGGCGGAACGGCGCTGGCAAGACCAATTGCCTGGAGGCGATCTCGCTGTTGTCGCCGGGCGCGCG
TTGGCCCGCGCACGCTCGACGACATCGCCGACAACCATGGCGATGGCTCCTGGGCGGTCTCCGCCGAAGTGAAGGGCGG
CTCGGCTCGCCACGCTCGGCACCGGCATCGATCCGGCACCGAGGCCCGC
>seq96
ATGTACATCGCTCCTCGAACAGCAGCGCCGGCAATAGCTCCATCAAAGCCAGCGCGGACCAACCGCGCGCGCAGAAG
GCCGGCAGCGCGGTCCGACGGCCCGGCCAACGACGAGCTTCGTCGAAAGCCGACGCCAACCCGGCGACCGAGATCGAA
ATATCGAGCTACGCCCGCGCTTATTGGCACGTGCCAGCGCGAGCAAGCCGTCGTGGCGCAGCTGTTGGCGCAGATCAAC
GCGCTCCGGCACGGCAGCTCGTGGTCCGGCCCGGAGCGGCCCGGCGACTACGCCACGA

```

Figure 3.6: Clusters file

2. Get a sequence which gives highest alignment score
3. Get a cluster represented by the sequence from the previous step
4. Make an alignment of the sequence to the cluster

The first step is to make an alignment of the sequence to the centroids. Since the alignment of the sequence to the two different centroids is independent, to make process faster, we can run it in parallel. Due to the parallel running, we add a preprocessing step which divides centroids into partitions. Afterwards, all other steps are run on the each partition.

3.4. Final design

We were tested our approaches according to our two requirements:

1. To be able to create a reduced dataset from a big dataset (more than 100000 sequences)
2. To make faster an identification of a new sequence

It has been shown that Clustering using UCLUST fits the best to our requirements, while the Hierarchical clustering using multiple alignment fails on the first requirement, the Single-link clustering fails on the second one. According to that, our final solution is implementation of the Clustering using UCLUST.

The implementation can be found on the link <https://github.com/Dorija/clust>. The implementation is made using C++, standard C++11. It is created for Linux operating system. We created two separated programs, the program for creating a reduced dataset called Reducer and the program to identified a new sequence called Searcher. The Reducer and the Searcher follow the procedures described in the section 3.3. Reducer options can be found in table 3.3, while in table 3.4 are options for Searcher.

Requirements for both programs:

- C++ compiler which supports standard C++11/C++0x
- Usearch package (<http://www.drive5.com/usearch/download.html>)
- OpenMP (supported by the GNU Compiler Collection) - optional

To run the Reducer properly, Usearch has to be in you \$PATH. The program can be run successfully even without OpenMP support. Without the support, all the work which can be run in parallel, will be executed in serial.

The number of threads used for parallel sections are correlated with the characteristics of the user's computer.

Table 3.3: Options Reducer

Option	Short form	Description	Mandatory
-in	-i	Input dataset - FASTA format	1
-out	-o	Output file for reduced dataset (centroids) - FASTA format. Default is "centroids.txt"	0
-clusters	-c	Output file for clusters. Default is "clusters.txt"	0
-identity	-p	Identity threshold, default: 0.5	0

To use the Reducer, first create a binary executable file. The file can be created typing a command:

```
g++ -std=c++0x -fopenmp src/reducer.cpp -o Reducer
```

To create a binary executable file for the Searcher, type a command:

```
g++ -std=c++0x src/searcher.cpp -o Searcher
```

Table 3.4: Options Searcher

Option	Short form	Description	Mandatory
-in	-i	Input sequences (one or more) which has to be found - FASTA format	1
-dataset	-d	Input file which contains reduced dataset (centroids) - FASTA format	1
-clusters	-c	Input file which constrains clusters	1
-identity	-p	Identity threshold for searching, default: 0.5	0

The output of the Reducer are two files, the file with centroids (reduced database) and the file with clusters. The Searcher prints the output on the standard output. First it prints the name of the sequence which we want to identify followed by colon (:) and the list of all found sequences.

3.4.1. Example

Here is a example of running the Reducer and the Searcher. Please go through an example before running on you own data.

Fist run the Reducer on test.fasta:

```
./Reducer --in data/test.fasta --out centroids.fasta  
--clusters cluster.txt --identity 0.9
```

To identify a new sequence (seq.fasta):

```
./Searcher --in data/seq.fasta --dataset centroids.fasta  
--clusters cluster.txt --identity 0.9
```

The output should be as follows:

```
test1:  
  seq32  
test2:  
  seq24
```


4. Results

Our approaches were tested on a set of bacterial coding sequences (CDS). Since the tests were made on different sizes of datasets, the names and the sizes of the datasets can be found in table 4.1. A unit for the size of the dataset is a number of sequences. The *Small*, *Medium* and *Big* datasets are obtained from *Bact_cds* dataset, by randomly choosing a specific number of sequences. The dataset *Bact_cds* is generated using bacterial genomes obtained from NCBI. Current number of available bacterial genomes on NCBI is 6385. Genes are extracted from bacterial genomes, and they are eventually used to generate our *Bact_cds* dataset. A length distribution of the sequences from *Bact_cds* dataset is shown in figure 4.1. By the length, we consider a number of nucleotides. On the horizontal axis are shown lengths of sequences, and on the vertical axis frequency. The longest sequence has 110418 nucleotides, while the shortest one has 18. Since the most significant concentration of the sequence length is in the first few bins, it is interesting to see a distribution inside those bins. The distribution is shown in figure 4.2.

To test our method, as a measure we use *accuracy*. Accuracy is defined as a percentage of the correct identifications from the reduced database. The reduced database is created, and afterwards we search through our reduced database for the sequences from the original database. If the sequence which we are using for searching is the same as the one found, the identification is correct.

Table 4.1: Dataset

Dataset	Size
Bact_cds	8726645
Small	150
Medium	1500
Big	50000

The results for the UCLUST clustering are shown in tables 4.2 and 4.3. Since there

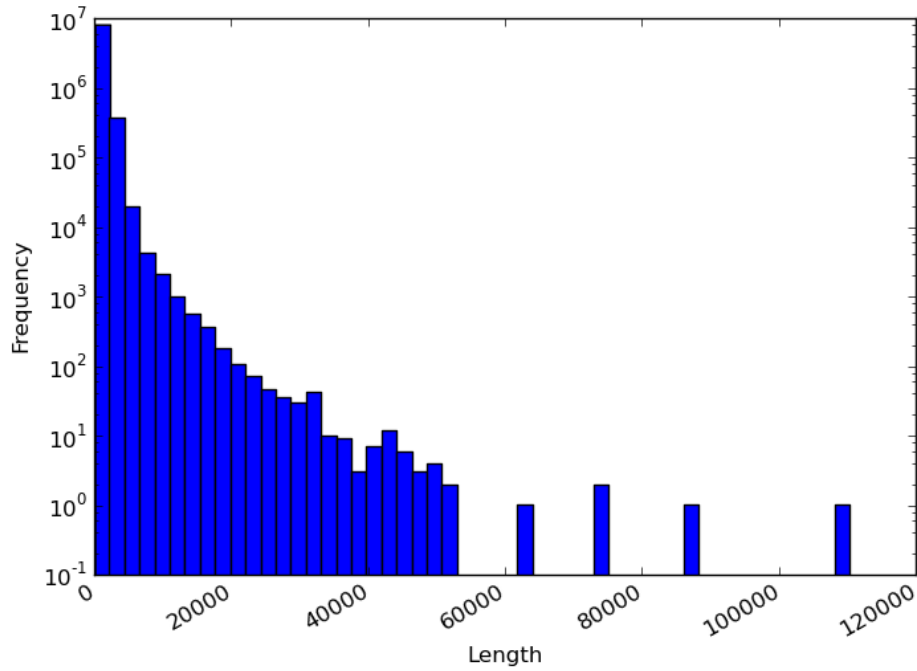


Figure 4.1: Distribution of the sequence length

is no limitation on the size of the database, the method is run using all of our datasets (*Small*, *Medium*, *Big*, and the original one *Bact_cds*). The results are shown for the different datasets and the parameter T , identity threshold.

In 4.2 are shown accuracy and average time to identify one sequence. To measure accuracy, we ran the Searcher with the sequences from a specific database, dataset and clusters, which were given to Searcher, were the ones created with the Reducer for the same database. For the *Small* and *Medium* dataset, we have used all of the sequences from the dataset for identification with the Searcher. Since running the Searcher for the *Big* and *Bact_cds* is more exhaustive, and datasets consist of much more sequences than the first two datasets, for the purpose of testing we have to randomly choose some smaller set of the sequences. We pick up 1000 sequences from the *Big* and *Bact_cds*. Time is representing average time for an identification of one sequence. The Searcher has to do a preprocessing of a sequence, before the sequence is ready to be identify. Time for an identification starts running only when the sequence is ready.

The sizes of the data sets are shown in table 4.3. Column Percentage stands as a percentage of a reduction. A reduction is calculated as a number of the sequences from the original dataset, which are excluded from the reduced dataset. According to

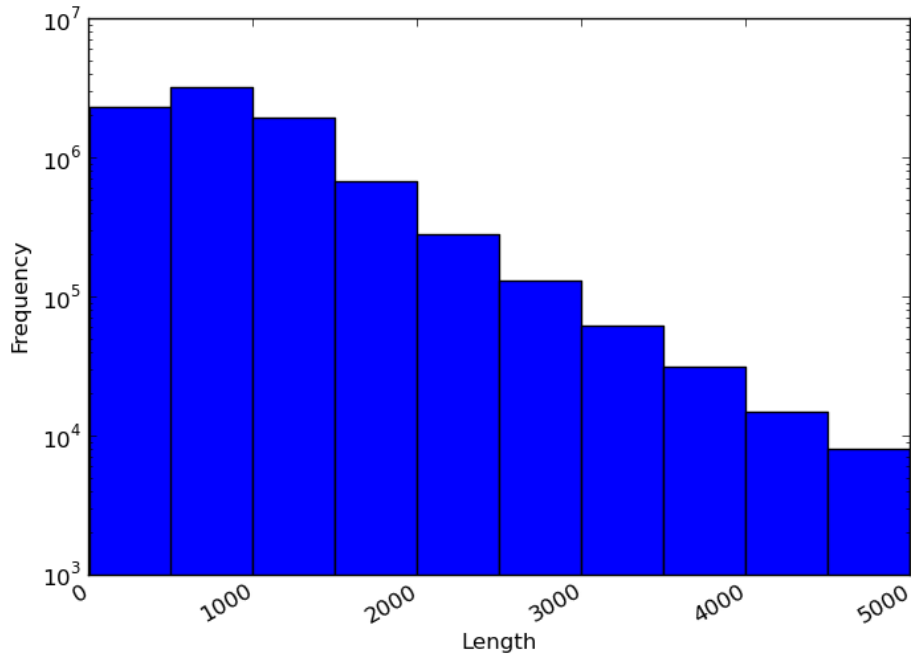


Figure 4.2: Distribution of the sequence length

the results, for a smaller threshold percentage of a reduction is larger.

The difference in the threshold value are reflected not only on the percentage of a reduction, but also on the accuracy of identification. In general, we can conclude that higher threshold results in higher accuracy.

Even though the percentage of a reduction is larger for threshold value 0.5 than for values 0.7 and 0.9, we can observe that average time on datasets the *Small* and *Medium* is smaller for values 0.7 and 0.9. The reason for the behavior is that the size of the datasets is not large enough to to achieve full potential of the clustering. The smaller datasets, even though they can be reduced, do not have ability to overcome the additional computational cost of managing the clusters. A real potential of the clustering is achieved if the computational cost of managing the clusters is less than the computational cost of going through the sequences which would be excluded from the Reduced dataset.

Table 4.2: Accuracy and time

Dataset	Threshold	Accuracy	Time(s)
Small	0.9	100%	0.00040
Small	0.7	100%	0.00047
Small	0.5	100%	0.00120
Medium	0.9	100%	0.00142
Medium	0.7	99.86%	0.00148
Medium	0.5	96.66%	0.00213
Big	0.9	99.90%	0.03985
Big	0.7	99.90%	0.04039
Big	0.5	93.50%	0.00310
Bact_cds	0.9	95.60%	5.38980
Bact_cds	0.7	81.50%	16.08140
Bact_cds	0.5	70.81%	9.06102

Table 4.3: Percentage of a reduction

Dataset	Threshold	Original size	Reduced size	Percentage
Small	0.9	150	150	0%
Small	0.7	150	150	0%
Small	0.5	150	24	84%
Medium	0.9	1500	1488	0.80%
Medium	0.7	1500	1478	1.47%
Medium	0.5	1500	106	92.93%
Big	0.9	50000	48907	2.19%
Big	0.7	50000	44743	10.57%
Big	0.5	50000	26093	47.82%
Bact_cds	0.9	8726645	5284712	40%
Bact_cds	0.7	8726645	4114545	52.25%
Bact_cds	0.5	8726645	1758319	79.86%

5. Conclusion

For the purpose of a faster identifications of sequences, the current database should be modeled in a different way. One of the possible ways is to grouping a sequences which are similar to each other, and represent them with one of the significant sequence. The significant sequences are part of the reduced database, while all the other sequences are hidden behind.

We proposed three different approaches for creating reduced database. The single-link clustering was the first try. It usually produced very unbalanced dendrograms. Since unbalanced dendrograms can not decrease searching time, we found the method inappropriate and continued with different approach. The hierarchical clustering with multiple alignment has showed up as a very suitable considering the accuracy, but the problem was a spread when searching for a new sequence. For the hierarchical clustering method we had choose a HMM profiles to represent each cluster. One of the limitation of the method was limitation of the current multiple alignment softwares. We were aware that accuracy of multiple alignments software for a huge dataset can be questionable, but we wanted to try our method for that kind of dataset. The time was a problem, hence we moved to the third approach. The third and the last approach involved partitional clustering. The idea behind was to group the sequences into partition where each partition consists of the sequences with similarity between them more than some given number. We ended up with each partition represented with one sequence which is called centroid and all the other sequences in the partition has identity between centroid larger than some given number.

Since the third approaches has been shown as the best, our final design contains implementation of it. Two programs has been produced, the Reducer and the Searcher. The Reducer creates a reduced database, while the Serchers search through the reduced database to find a new sequence. To create a reduced database, different threshold value for identity can be given. According to our results, the threshold value should be correleted with the initial size of the database and the needs of the user. Accuracy of the Searcher is higher, if the threshold value is higher. The cost of accuracy is usually

time, even though we have shown that it is not the case for initially smaller dataset.

There is always a better way, hence for the future work we would like to try again with a hierarchical clustering and multiple alignment, and to try with different way of representing a clusters.

BIBLIOGRAPHY

- Löytynoja A. and Goldman N. An algorithm for progressive multiple alignment of sequences with insertions. *PNAS*, 102, 205.
- Richard C. Duber Anil K. Jain. *Algorithms for Clustering Data*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA ©1988, 1988.
- BLAST® Command Line Applications User Manual*. Bethesda (MD): National Center for Biotechnology Information (US), 2008. URL <http://www.ncbi.nlm.nih.gov/books/NBK279690/>.
- Sean R. Eddy. Profile hidden markov models. *Bioinformatics*, 14, 1998.
- Sean R. Eddy and Travis J. Wheeler. *HMMER User's Guide*, 2015. URL <ftp://selab.janelia.org/pub/software/hmmer3/3.1b2/Userguide.pdf>.
- EMBOSS Smith-Waterman, 2015. URL http://www.ebi.ac.uk/Tools/psa/emboss_water/help/index-nucleotide.html.
- Sievers F., Wilm A., Dineen DG., Gibson TJ., Karplus K., Li W., Lopez R., McWilliam H., Remmert M., Söding J., Thompson JD., and Higgins DG. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7, 2011.
- Profile HMM Analysis. Profile hidden markov models analysis, 2015. URL <http://www.biology.wustl.edu/gcg/hmmanalysis.html>.
- Edgar RC. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26, 2010.
- Simossis V., Kleinjung J., and Heringa J. An overview of multiple sequence alignment. *Current Protocols in Bioinformatics*, 3, 2003.

Reducirana baza gena za precizno određivanje vrsta

Sažetak

U svrhu brze identifikacije vrsta, trenutne baze podataka bi se trebale remodelirati. Trenutne baze sadrže sekvence koje su slične jedne drugima, što dovodi do redundancije prilikom određivanja vrsta te usporava cijeli proces. Prijedlog je da se baze podataka konvertiraju u takozvane Reducirane baze podataka. Reducirana baza podataka sadržavala bi informacije o značajnim sekvencama, a sve ostale sekvence bile bi skrivene iza za njih značajnih sekvenci. U ovom radu predložena su tri različita pristupa. Dva bazirana na hijerarhijskom grupiranju i jedan na partijskom grupiranju. Konačaj dizajn rada sadrži implementaciju pristupa baziranog na partijskom grupiranju, budući da se taj pristup pokazao kao najbolji.

Ključne riječi: Grupiranje, Redukcija genoma, Identifikacija

A reduced gene database for precision species detection

Abstract

For the purpose of a faster identification of sequences, the current database should be modeled in a different way. The current databases should be converted to the reduced database. The reduced database is modeled in a way that at first it gives us information only about significant sequences, and all the other sequences are hidden behind their representative. We proposed three different approaches for creating reduced database. Two of them incorporates hierarchical clustering, while the third one is based on partitional clustering. Since the approach with partitional clustering has been shown as the best, our final design contains implementation of it. Two programs has been created, the Reducer and the Searcher. The Reducer creates a reduced database, while the Serchers search through the reduced database to find a new sequence.

Keywords: Clustering, identification, Reduced database