Optimizing machine learning techniques for genomics clustering

Johny Matar

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UNIVERSITÉ DE
FRANCHE-COMTÉ

THÈSE DE DOCTORAT

DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

JOHNY MATAR

Optimizing machine learning techniques for genomics clustering

Optimisation des techniques d'apprentissage automatique pour le clustering génomique

Thèse présentée et soutenue à Belfort, le 16 decembre 2021

Composition du Jury :

PR SPITERI PIERRE	Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique, d'Hydraulique et des Télécommunications	Président
PR DARAZI RONY	Université Antonine	Rapporteur
DR VERNIER FLAVIEN	Université Savoie Mont Blanc	Rapporteur
PR GUYEUX CHRISTOPHE	Université Bourgogne Franche-Comté	Directeur de thèse
DR EL KHOURY HICHAM	Université Libanaise	Co-encadrant de thèse
DR CHARR JEAN-CLAUDE	Université Bourgogne Franche-Comté	Co-encadrant de thèse

ABSTRACT

Optimizing machine learning techniques for genomics clustering

Johny Matar
University of Bourgogne Franche Comté, 2021

Supervisors: Christophe Guyeux, Jean-Claude Charr, and Hicham El Khoury

In the field of bioinformatics, clustering recently appeared to be a very efficient technique for sequence analysis. While greedy and hierarchical algorithms are used in the majority of the available tools, spectral clustering was recently introduced as a new stakeholder in this field. Spectral clustering is an efficient technique for well-separated sequence clustering and GMM's (Gaussian Mixture Models) are often able to cluster overlapping groups given an adequately designed embedding. Yet, the available clustering tools, for biological sequences, present many drawbacks especially that i- the most widely used ones require an accurate choice of a non-obvious identity or similarity threshold, ii- most of them are not designed to cluster potentially divergent sequences, and iii- the recent one that relies on the spectral clustering technique, and that does not require any user intervention or prior knowledge about the input sequences, is so slow and was not enough tested and validated. Moreover, the performance of several well-known clustering techniques is not assessed in the field of clustering biological sequences.

Firstly, since the recent clustering technique that relies on spectral clustering offered a potential solution for the drawbacks of the traditional tools, its own drawbacks are being addressed and an enhancement in its computation time is achieved. This enhancement is based on improving the required time for the pairwise affinity computation of the sequences. The proposed solution is to adopt a parallel computation scheme for the pairwise affinity computation. This solution has been implemented according to the master/slave distributed architecture, using Message Passing Interface (MPI), and showed a drastic improvement in the computation time. Moreover, the resulting clustering package, named SpCLUST, was intensively evaluated on simulated and real genomic and protein data sets. The clustering results were compared to the most known traditional tools, such

as UCLUST, CD-HIT, and DNACLUST. The comparison showed that SpCLUST outperforms the other tools when clustering divergent sequences.

Secondly, further improvements to SpCLUST, speed-wise, accuracy-wise, and feature-wise, were introduced. The implemented approach in SpCLUST results in a pipeline of the following steps: i- sequence alignment, ii- pairwise affinity computation of the sequences, iii- Laplacian Eigenmap embedding of the data, and iv- GMM based clustering. Therefore, improving the quality of the generated clustering and the performance of this approach is directly related to the enhancement of each of these five steps: the alignment quality, the appropriate design of the affinity, the GMM implementation, etc. Accordingly, we have written a completely new C++ GMM library incorporating new features and options for optimizing the clustering speed and quality. This resulted in a second release, namely SpCLUST-V2, of our package. Moreover, the impact of using different modules, methods, implementations, and algorithms (sequence alignment modules, various clustering methods, GMM implementations, and affinity matrix types) in this process pipeline is carefully discussed.

Finally, a major improvement in the speed of the pairwise affinity computation is achieved by adopting a new library in our package. Moreover, a novel clustering technique is introduced. Furthermore, additional clustering techniques were explored on biological sequences, and a qualitative study compares their performance and accuracy. The used implementations were embedded in SpCLUST-Global, an improved cross-platform biological sequences' clustering package. SpCLUST-Global outperforms its GMM-based predecessors in terms of speed and handling data sets that contain large genomes. It also outperforms the state-of-the-art tools in clustering hybrid and highly divergent data sets. The versions of our package are freely available online.

KEYWORDS: Biological sequences clustering, Genomics, Laplacian Eigenmaps, Gaussian Mixture Model, Parallel computation, Spectral clustering, Clustering quality analysis, CHAINS clustering, MOTIFS-based spectral clustering, Affinity matrices, Sequences alignment.

RÉSUMÉ

Optimisation des techniques d'apprentissage automatique pour le clustering génomique

Johny Matar
Université de Bourgogne Franche Comté, 2021

Encadrants: Christophe Guyeux, Jean-Claude Charr et Hicham El Khoury

Dans le domaine de la bioinformatique, le clustering est une technique efficace pour l'analyse des séquences. Le clustering spectral a récemment été introduit comme un nouvel acteur dans ce domaine. C'est une technique efficace pour le clustering de séquences bien séparées et les GMM sont souvent capables de partitionner des groupes qui intersectent. Pourtant, les outils de clustering disponibles, pour les séquences biologiques, présentent de nombreux obstacles: i- les plus utilisés nécessitent un choix précis d'un seuil d'identité ou de similarité qui n'est pas toujours évident, ii- la plupart d'entre eux ne sont pas conçus pour regrouper des séquences assez divergentes, et iii- une technique récente, qui repose sur le clustering spectral, et qui ne nécessite aucune connaissance préalable des propriétés des séquences d'entrée, est assez lente et n'a pas été suffisamment validée. De plus, les performances de plusieurs techniques de clustering bien connues ne sont toujours pas évaluées dans le domaine du clustering de séquences biologiques.

Tout d'abord, étant donné que la technique récente qui repose sur le clustering spectral offre une solution aux obstacles connus des outils traditionnels, des solutions à ses propres obstacles seront visée. Cette amélioration est basée sur la réduction du temps requis pour le calcul d'affinité par paires de séquences. La solution proposée est d'adopter un schéma de calcul parallèle pour ce calcul. Cette solution a été implémentée, selon l'architecture distribuée maître/esclave, en utilisant la MPI, et a montré une amélioration considérable du temps de calcul. De plus, l'outil de clustering résultant, nommé SpCLUST, a été intensivement évalué sur des ensembles de données génomiques et protéiques. Les résultats du clustering ont été comparés à celui des outils traditionnels

les plus connus, tels que UCLUST, CD-HIT et DNACLUST. La comparaison a montré que SpCLUST surpasse les autres outils lors du regroupement de séquences divergentes.

Ensuite, d'autres améliorations de SpCLUST, en termes de vitesse, de précision et de fonctionnalités, ont été introduites. L'approche implémentée dans SpCLUST consiste des étapes suivantes : i- alignement de séquences, ii- calcul d'affinité par paires de séquences, iii- intégration des données sur la Eigenmap laplacienne et iv- clustering basé sur GMM. Par conséquent, l'amélioration de la qualité du clustering généré et des performances de cette approche est directement liée à l'amélioration de la qualité de l'alignement, la conception appropriée de l'affinité, l'implémentation GMM, etc. En conséquence, nous avons écrit une bibliothèque GMM intégrant de nouvelles fonctionnalités et options pour optimiser la vitesse et la qualité du clustering. Cela a abouti à une deuxième version de notre outil, nommée SpCLUST-V2. De plus, l'impact de l'utilisation de différents modules, méthodes, implémentations et algorithmes dans ce pipeline de processus est soigneusement discuté.

Enfin, une accéleration majeure de la vitesse du calcul d'affinité par paire est obtenue en adoptant une nouvelle bibliothèque dans notre package. De plus, une nouvelle technique de clustering est introduite. Aussi, des techniques de clustering supplémentaires ont été explorées sur des séquences biologiques, et une étude qualitative est présentée pour leurs résultats. Ces résultats sont également comparés à ceux de certains outils traditionnels. Les implémentations utilisées ont été intégrées dans SpCLUST-Global, un outil amélioré de regroupement de séquences biologiques multiplateformes. SpCLUST-Global surpasse ses prédécesseurs qui sont basés sur GMM, en termes de vitesse et de gestion des ensembles de données contenant de grands génomes. Il surpasse également les outils traditionnels en termes de justesse de regroupement d'ensembles de données hybrides et très divergents. Les différentes versions de notre outil sont disponibles gratuitement en ligne.

Mots clés: Clustering de séquences biologiques, Génomique, Eigenmaps laplaciennes, Modèle de mélange gaussien, Calcul parallèle, Clustering spectral, Analyse de qualité de clustering, Clustering CHAINS, Clustering spectral basé sur des MOTIFS, Matrices d'affinité, Alignement de séquences.

ACKNOWLEDGEMENTS

My sincere gratitude and immeasurable appreciation go to the following persons who have contributed, in a way or another, in directing the path for my doctoral degree to the present end.

Prof. Christophe Guyeux, my thesis director, for believing in my capabilities, proposing the interesting topic of my research, and giving me the opportunity to start this path.

Dr. Jean-Claude Charr, my thesis co-director at UBFC, for his priceless remarks and constructive criticisms that lead to great enhancements in the presentation of the results and the manuscripts of my publication.

Dr. Hicham El Khoury, my thesis co-director at UL, for the close follow-up that he provided during my stays in Lebanon. His continuous motivation and support was essential, especially during the deep economic crisis that hit our country and that resulted in a devastating impact on all levels.

Prof. Stéphane Chrétien, for his valuable propositions and interventions that contributed in the enhancement of my research.

My close friends, and my family members, for their support and encouragement to keep going forward despite the unexpected and hard challenges that arise.

The contributors to the GISAID, NCBI, and VIRUSITE databases from which were retrieved the sequences used in this thesis.

Finally, I would like to express my deepest thanks to the **jury members** for accepting to be reviewers and examiners of this thesis.

CONTENTS

List of abbreviations

AIC . . . . . Akaike Information Criterion

ARI . . . . . Adjusted Rand Index

BIC . . . . . Bayesian Information Criterion

BLOSUM . . . . BLOcks SUbstitution Matrix

CH . . . . . Calinsky-Harabasz

DB . . . . . Davies-Bouldin

DBSCAN . . . . Density-Based Spatial Clustering of Applications with Noise

DNA . . . . . DeoxyriboNucleic Acid

DP-Means . . . Dirichlet Process Means

EM . . . . . Expectation Model

GMM . . . . . Gaussian Mixture Model

HDBSCAN . . . Hierarchical DBSCAN

HIV . . . . . Human Immunodeficiency Virus

MERS . . . . . Middle Eastern Respiratory Syndrome

ML . . . . . Machine Learning

MPI . . . . . Message Passing Interface

NADH . . . . . Nicotinamide Adenine Dinucleotide Hydrogen

OTU . . . . . Operational Taxonomic Unit

PAM . . . . . Point Accepted Mutation matrix

RAM . . . . . Random Access Memory

RNA . . . . . RiboNucleic Acid

rRNA . . . . . Ribosomal RNA

RWNL . . . . . Random Walk Normalized Laplacian

SARS . . . . . Severe Acute Respiratory Syndrome

SARS-COV . . . SARS - COrona Virus

I

DISSERTATION INTRODUCTION

1

INTRODUCTION

The Machine Learning (ML) techniques in general, and the Expectation Maximization (EM) models [Dempster et al., 1977, Wu, 1983, McLachlan et al., 2007] in particular, are playing a paramount role in a wide variety of applications and data analysis. The scope of usage of EM includes the analysis of networks traffic [McGregor et al., 2004], agricultural production systems [Liakos et al., 2018], and medical devices production such as the hand prothesis [Paaßen et al., 2018], among others. Yet, the wide majority of the tools targeting the analysis of biological sequences, especially the clustering ones, still rely on the classical greedy algorithms. Therefore, this thesis focuses on advancing the state of the art of clustering biological sequences, with a specific focus on optimizing machine learning techniques for this purpose, handling potentially divergent sequences, and introducing novel techniques in this field.

According to the convention of co-direction between the Bourgogne Franche-Comté university (UBFC) and the Lebanese University (UL), the presented research was carried out in alternation between the FEMTO-ST laboratory in Belfort, France, and the LaRRIS laboratory in Fanar, Lebanon. Throughout the remainder of this thesis, the writer will be referred to as “we”, rather than “I”. This is because this thesis presents research performed in a collaborative setting, as part of the teams in both research laboratories.

This chapter provides an introduction to the work done in this thesis. It addresses the general context and the relevant problematics in the considered use cases. Then, it presents briefly the contributions of this thesis.

1.1/ INTRODUCTION TO BIOLOGICAL SEQUENCES CLUSTERING

Sequence clustering refers to the act of partitioning an input group of sequences into clusters, each containing a group of somehow related sequences. It can involve either nucleotide or protein based sequences and is mainly used to identify sequences that are potentially derived, by mutations or substitutions, from each others or from a common

ancestor. Indeed, the clustering of biological sequences is currently playing a paramount role in the analysis of the biological sequences, by linking the huge number of newly discovered sequences to their variants and ancestors.

One of the methods for grouping the related sequences is using some tools or algorithms for building phylogenetic trees [Stamatakis, 2014, Guindon et al., 2009, Lefort et al., 2017, Ronquist et al., 2012] when the evolution is assumed to be in that form. But technically, this method might fail in some cases: for example, if the analyzed data set contains two distinct populations of bacteria, where one of these populations have received some genes from the other population by horizontal gene transfer, then the phylogenetic signal will be scrambled and lead to a poorly resolved tree. In this case, the strains having received these genes will be wrongly positioned in the tree, while a well done grouping should split them into two groups.

In addition, other methods resulted from a great deal of work that has been done in this field, and many tools were made for this specific task. In fact, many clustering tools, targeting a fast clustering of highly similar sequences, exist [Li et al., 2006, Edgar, 2010, Ghodsi et al., 2011, Mercier et al., 2013, Jiang et al., 2016, Matias Rodrigues et al., 2013]. Moreover, a recent tool that targets the clustering of potentially divergent sequences [Bruneau et al., 2018], have been published. Yet, none of the existing clustering tools can claim its clustering superiority in all possible cases and on all kind of sequences.

The purpose of this thesis is therefore to suggest solutions that enhance the analysis of the biological sequences. The contributions concentrate specifically on enhancing the sequences clustering part, that attracted a large number of researchers. These researchers presented many clustering approaches to perform this particular task, and the implementations of these approaches contributed in the release of a large variety of clustering tools.

1.2/ PROBLEMATICS WITH REGARDS TO THE USE CASES

The mutations, substitutions, and gene transfers occur upon the duplications of biological sequences, e.g., viruses and bacteria, for many reasons [Duffy et al., 2008, Wielgoss et al., 2013, Oliver et al., 2010]. These changes result in a continuous increase in the number of the newly discovered sequences. In addition, these types of changes occur in an unpredictable degree. Therefore, linking these discovered sequences to their siblings or ancestors, is not an obvious task, especially that the arising degree of divergence is usually unknown, and may be greater to the theoretical expectations.

Therefore, the work done in this thesis addresses numerous limitations and issues in

the framework of biological sequences analysis in general, and in its clustering part in particular. These limitations and issues start from the need for a non-obvious similarity threshold parameter, for using the traditional tools that target the clustering of highly similar sequences. It extends to the computation complexity and the lack of intensive validation for the introduced technique, in the first of its kind of tool that targets the potentially divergent sequences.

While the widely used and largely tested clustering tools are those targeting the highly similar sequences, their efficiency in clustering sequences that show a high level of divergence requires to be investigated. Their need for a user-defined similarity or identity threshold is also to be tackled. The lack of accuracy in the choice of this threshold, can be highly relevant in the case of handling sets of newly discovered sequences.

Moreover, the tool that targets the potentially divergent sequences promotes the use of an intervention-free technique that is new in this field, namely the Gaussian Mixture Model (GMM) [McLachlan et al., 2004] that is calibrated by an Expectation Maximization (EM) model. This tool also computes a global alignment for the input sequences, then it uses the aligned sequences to compute a pairwise similarity between the input sequences. Therefore, this scheme requires an intensive computation when a large number of sequences is involved, and raises a very challenging speed optimization problem.

1.3/ MAIN CONTRIBUTIONS

The main contributions in this dissertation fall within the optimization of the aforementioned clustering task for biological sequences. These contributions can be summarized as follows:

1. First, we propose a major performance optimization scheme for the GMM-based tool. This scheme is divided into three stages. In the first stage, we propose a method that is based on a parallel computation design for the pairwise similarity among the input sequences. In this method, the concerned module of this tool was ported to a lower level programming language, the C++, where the Message Passing Interface (MPI) was used to achieve the parallelism. This first improvement moved the bottleneck, in the GMM-based tool, to the GMM module itself. This issue was also addressed by porting this module to a lower level programming language. In the final stage of the performance optimization, a performant sequence alignment library was adopted. Its implementation substitutes a third party tool that was initially used to perform this task. After the implementation of each stage from this optimization scheme, the results of our tests showed a drastic decrease in the computation time that is required for each stage, while preserving a good quality of the

clustering results. These results validated the efficiency of our proposed scheme.

2. Second, we investigate the validity of using the Gaussian Mixture Model (GMM) for the clustering of biological sequences. The Expectation Model (EM), that was pre-defined and coupled with the GMM by the authors of [Bruneau et al., 2018], is applied and validated for biological sequences. For this purpose, several datasets from real biological sequences were assembled, in addition to several simulated datasets. These simulated sets reflect various degrees of divergence, and were generated by a tool that was specifically developed for this purpose. The real and the simulated datasets were used in an intensive validation process, for the GMM clustering technique in this field. The validity of the obtained clustering results, demonstrated the efficiency of using the EM-GMM for clustering biological sequences.

3. Alongside with the re-coding of the GMM module, that took place in the second stage of the performance optimization, the third contribution of this thesis is introduced. Three additional techniques, based on the GMM, are proposed and implemented in our GMM module. These techniques rely on an iterative process that modifies the initial and random distribution of the centroids of the clusters at each iteration. Since this modification leads to a potentially different result at the convergence of the GMM, then the best result can either be chosen via a quantitative approach such as the number of occurrences of the final grouping, or a qualitative approach such as finding the grouping that scores the best Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC) [Vrieze, 2012]. Moreover, the use of three additional types of affinity matrices[1] is made available in our implementation. The proposed techniques and the added types of affinity matrices are validated by the experiments that are made on a variety of datasets. The efficiency of these techniques is also validated in clustering hybrid datasets, in addition to clustering datasets of sequences where a horizontal gene transfer is simulated among the involved sequences.

4. In the fourth contribution, we validate the application of three additional state of the art clustering approaches, that were not yet used in the field of clustering biological sequences. Two of these approaches, the DBSCAN [Khan et al., 2014] and HDBSCAN [Campello et al., 2013], are not recent, conversely from the third one that is based on GMMs and uses a MOTIFS [Benson et al., 2016] scheme for clustering a graph. Therefore, applying and validating this third approach in clustering biological sequences, required proposing additional techniques, as part of this contribution, in order to build the initial graph. The validity of the clusterings that were produced by each approach, on datasets having different properties, are discussed. More-

[1] a transformation made to the initial similarity matrix prior the computation of the spectrum.

over, the proposed techniques that enabled the application of the third and recent MOTIFS-based approach were also validated.

5. Finally, we propose a novel, fast, and simple clustering approach. This approach was named CHAINS and uses the computed pairwise similarities, among the sequences, to perform the clustering directly. It was compared to all the previously tackled clustering approaches. In terms of clustering quality, and based on the results that were obtained from various experiments, our new approach outperformed its competitors in the vast majority of our experiments and use cases. Moreover, the results of the experiments show that the implementation of our new approach also outperforms its competitors, speed-wise, except for the traditional biological sequences clustering tools. The traditional tools remain the best suited for rapidly clustering highly similar sequences, where the similarity threshold is predictable.

1.4/ Dissertation Outline

The rest of this dissertation is organized as follow: Chapter 2 presents various state of the art clustering techniques. Chapter 3 discusses the main differences between the state of the art clustering tools, while highlighting their adopted clustering techniques. Chapter 4 presents and discusses various clustering validation indexes that will be later used for validating the results of our experiments. Chapter 5 presents the first set of contributions of this dissertation, precisely the first speed optimization stage of the initial GMM-based tool, and the intensive validation for using the GMM in clustering the biological sequences. Chapter 6 investigates the effect of using various novel GMM-based techniques, in addition to the achieved performance improvement in our GMM module. It further validates the use of the GMM approach that resolves the issue of requiring an accurate similarity threshold. Chapter 7 presents a comparative and qualitative study on various state of the art approaches that were not yet used in the field of clustering biological sequences. A novel and performant approach is also introduced in Chapter 7, and involved in the qualitative study. Finally, Chapter 8 concludes the work that has been done in this thesis.

II

Clustering: techniques, tools, and validation

This part examines the state of the art for the clustering techniques in general. Then it highlights the techniques that are used in the existing tools for clustering biological sequences, and the targeted use cases of these tools. Finally, it presents some state of the art indexes for assessing the quality of a given clustering. The first chapter of this section presents the clustering techniques. The second chapter introduces some existing tools for clustering biological sequences and highlights the differences between their algorithms. The third chapter discusses the validation techniques for the clustering.

2

CLUSTERING TECHNIQUES

Clustering is the act of partitioning a set of elements into groups of somehow related ones. A wide variety of clustering algorithms are proposed and discussed in the literature [Mirkin, 2012], and some of the most popular ones are presented in the following sections.

2.1/ HIERARCHICAL

A hierarchical [Murtagh et al., 2012] algorithm is considered as a greedy algorithm. It starts by considering each element as a singleton cluster in its agglomerative approach. The distances or similarities between the elements are then used in order to progressively merge the nearest neighboring clusters. The merging stops at the point where the resulting clusters are well separated by a distance greater than a certain threshold, or inversely a similarity smaller than a certain threshold. This threshold must be chosen prior to the application of this algorithm.

2.2/ K-MEANS

K-Means [Likas et al., 2003] is an algorithm that requires the input of K, the number of resulting clusters, before its application. It starts by arbitrary placing the centers of the clusters, then iteratively moving them until they reach their optimal position where spherical-shaped clusters cover all the elements. The Euclidean distances, between the clustered elements, are usually used in this process. In a very similar algorithm, called **DP-Means** [Kulis et al., 2011], a new cluster is created only when a point is sufficiently distant from all the other centers.

2.3/ GMM

The GMMs [Guo et al., 2012] are based on statistical machine learning. They use a probabilistic method to determine densities and detect clusters. The set of parameters that govern the GMMs are estimated by the Expectation Maximization (EM) algorithm. Therefore, theses two algorithms are tightly bound together as EM-GMM. The GMMs are widely used in spectral clustering. Although the EM-GMM is a computationally heavy algorithm, where studies are continuously improving its speed via parallel computation, it outperforms K-Means by its ability in detecting random-shaped clusters.

2.4/ DBSCAN

DBSCAN [Khan et al., 2014] is an algorithm that separates random-shaped clusters based on the spatial density of the elements in the plane. In addition to the distance between the elements, DBSCAN takes a distance ϵ and a minimum cluster size $minPts$ as parameters. It selects a random point and forms a cluster consisting of that point and its neighbors that are distanced less than ϵ from it. This process is repeated on the unclustered elements until none is left. The singletons are classified as noise. Several algorithms that extend DBSCAN are also discussed in [Khan et al., 2014], and aim to successfully cluster a data with different densities among its clusters.

2.5/ HDBSCAN

HDBSCAN [Campello et al., 2013, Malzer et al., 2020] is a Hierarchical DBSCAN that does not require an ϵ input and that solves the limitation of DBSCAN in the case of variable-density datasets. Similarly to DBSCAN, HDBSCAN takes a minimum cluster size $minPts$ as a parameter then builds a hierarchy tree. Starting from the root of that tree, HDBSCAN splits the children into clusters if they contain a number of elements greater than $minPts$. Conversely, the elements of a child that contains less than $minPts$ are treated as noise.

2.6/ SPECTRAL CLUSTERING

The spectral clustering technique [Von Luxburg, 2007] takes a pairwise similarity matrix as input and consists of a three-stages process: i- the computation of the affinity matrix from the input similarity matrix, ii- the dimensionality reduction of the affinity matrix, ii- the clustering. These stages will be detailed in the following sub-sections.

2.6.1/ THE AFFINITY MATRIX

In the spectral clustering pipeline, the affinity matrix is usually computed as a Random Walk Normalized Laplacian. However, other interesting matrices have been proposed [Von Luxburg, 2007, Langone et al., 2011, Saade et al., 2014, Dall'Amico et al., 2019a], such as the Non-normalized Laplacian, Modularity [Langone et al., 2011] and the Bethe Hessian (Deformed Laplacian) [Dall'Amico et al., 2019b]. These matrices are defined as follows:

- **Non-normalized Laplacian:**

$$L = D - A,$$

  where A is the adjacency matrix between the sequences and D is its diagonal matrix of degrees.

- **Random Walk Normalized Laplacian:**

$$L^{rw} = D^{-1}L,$$

  where D is the degrees matrix of the adjacency matrix and L is the Non-normalized Laplacian matrix.

- **Modularity:**

$$M = \frac{1}{K}\left(A - \frac{1}{K}kk^T\right),$$

  where A is the adjacency matrix, k is the degrees vector of A, and K is the total degree of A.

- **Bethe Hessian:**

$$H_r = (r^2 - 1)I + D - rA$$

  where I is the identity matrix, D is the degrees matrix of the adjacency matrix A, and the constant r is the square root of the average degree of the graph, as suggested in [Saade et al., 2014].

Let us remark the following points concerning these definitions. The Laplacian is a symmetric and positive semidefinite matrix. The Non-normalized Laplacian and the Normalized Laplacian serve respectively in the approximation of the minimization of the RatioCut and the NCut. The Modularity is a quality function whose high values reveal the possible existence of strong communities. Finally, the Bethe Hessian, also called deformed Laplacian, features a regulator constant r in addition to the previously introduced Laplacian matrices.

2.6.2/ THE DIMENSIONALITY REDUCTION

The spectrum, from which derives the nomination of spectral clustering, is used to achieve the dimensionality reduction of the affinity matrix. This process consists in: i- computing the spectrum (eigenvalues) for the square affinity matrix of dimension $m \times m$, ii- selecting a certain number n of the smallest eigenvalues, iii- extracting their associated eigenvectors as being the most significant ones, iv- using the matrix formed from the most significant eigenvectors to get the coordinates of the data points in an n-dimensional plane. The number of the smallest eigenvalues, n, can be selected by using different techniques, such as the log or the $delta$. The log technique consists in selecting the $log(m)$ smallest eigenvalues, while the $delta$ technique stops the selection at the point where the difference δ, between the next smallest eigenvalue and the current one, becomes greater than a certain user defined value.

2.6.3/ THE CLUSTERING

The clustering part in the spectral clustering is achieved by using K-Means or GMMs, although any one of the aforementioned clustering techniques can be technically used. In order to use DBSCAN or HDBSCAN for example, a simple modification may be required to the Euclidean[1] or Manhattan[2] distance calculator, in order to compute this distance in an n-dimensional space according to the one provided in the extracted spectrum after the dimension reduction.

2.7/ MOTIFS-BASED CLUSTERING

A recent and novel clustering method, for higher-order networks, introduced in [Benson et al., 2016], achieves the clustering of a graph by performing cuts at the areas showing a minimum conductance. The conductance is the ratio of the number of cut MOTIFS, that are identified on the graph, to the smaller number of nodes that are involved in the MOTIFS, between either sides of the cut. The MOTIF can be chosen from a set of 13 different triangular shapes with directed sides. But since the computation of the conductance minimization is NP-hard, the authors in [Benson et al., 2016] proposed a computationally efficient solution of two stages: i- computing the matrix holding the number of MOTIFS in which each link is involved, i.e. the element (i, j) is the number of identified MOTIFS containing the link between the nodes i and j, ii- applying spectral clustering to this matrix. In [Benson et al., 2016], it was proved that the clustering that was obtained in stage ii matched the clustering that reflects the optimal cuts on the initial

[1] calculated as the square root of the sum of the squares of the differences of the coordinates.
[2] calculated as the sum of the absolute values of the differences of the coordinates.

graph, i.e., the cuts showing a minimum conductance. Therefore, the normalized Laplacian of this MOTIFS-based matrix can be considered as a new type of affinity computation for the spectral clustering.

2.8/ CONCLUSION

The presented clustering algorithms reflect major differences. An interesting one of these differences is the shape of the detected cluster that can be spherical-shaped or random-shaped for example. Moreover, the computation complexity and intensiveness represents another major aspect of difference among these algorithms. Therefore, their application fields shall be carefully selected based on these aspects of difference. Accordingly, the properties of some state of the art clustering tools for biological sequences will be discussed in the next chapter.

3

BIOLOGICAL SEQUENCES CLUSTERING TOOLS

Several packages for high speed clustering of nucleotide and/or protein sequences are publicly available, such as CD-HIT [Li et al., 2006], UCLUST [Edgar, 2010], DNACLUST [Ghodsi et al., 2011] and SUMACLUST [Mercier et al., 2013]. All these packages rely on greedy algorithms for clustering the sequences. Conversely, DACE [Jiang et al., 2016] relies on the scalable Dirichlet Process means (DP-means) algorithm, making it an algorithm similar to K-means. HPC-CLUST [Matias Rodrigues et al., 2013], for its part, is based on a hierarchical algorithm. Conversely, GCLUST [Bruneau et al., 2018] and AncestralClust [Pipes et al., 2021] are, to our knowledge, the only two released packages that target potetially divergent sequences. These tools play a vital role in the sequence analysis because the use of taxonomy dependent algorithms strongly rely on the completeness of existing databases, while the majority of newly discovered sequences are unknown. The main features and characteristics of these tools are presented the next sections.

3.1/ CLUSTERING HIGHLY SIMILAR SEQUENCES

3.1.1/ CD-HIT

CD-HIT is a suite of tools for biological sequence handling, including modules for clustering both nucleotide and protein sequences using word counting for computing the similarities, in order to avoid costly pairwise sequences alignments. It processes the input sequences by their order of length, starting from the longest and considers the first one as the first cluster representative. It then classifies the following sequences subsequently, based on the input similarity threshold, as either a new cluster representative or part of a cluster for a previously classified representative. CD-HIT is very fast and can handle extremely large databases [Li et al., 2006].

3.1.2/ UCLUST

UCLUST uses a module named USEARCH for assigning the sequences to their clusters, based on a given identity threshold and, optionally, an input of the centroid units. In contrast with CD-HIT, UCLUST does not sort the input sequences by length prior to clustering them, thus the order of the sequences can impact the result since most clusters representatives (or centroids) are chosen from the first sequences. UCLUST can also cluster both nucleotide and protein sequences, and is able to produce a better clustering quality than its competitor CD-HIT [Edgar, 2010] while consuming less memory.

3.1.3/ DNACLUST

DNACLUST was designed for quick clustering of highly similar DNA sequences, but it does not handle protein sequences. Similarly to CD-HIT, it first sorts the sequences in their decreasing order of length. If the sequences have equal length, they are sorted in their decreasing order of abundance. The first sequence is considered as the current cluster centroid and all the input sequences having a distance with the centroid inferior than the user input distance threshold are added to that cluster. This procedure is repeated until all the sequences are clustered. Based on its authors study, DNACLUST outperforms UCLUST when high similarity thresholds, above 0.95, are chosen.

3.1.4/ SUMACLUST

SUMACLUST, along with SUMATRA, is a package aiming for a fast and exact DNA sequences comparison. It first compares the pairwise similarities between sequences using SUMATRA, then sorts the sequences by abundance, and finally it clusters the sequences with a greedy algorithm similar to CD-HIT and UCLUST. The main difference between SUMACLUST and its competitors, CD-HIT and UCLUST, is that it uses a pairwise sequence alignment algorithm before clustering the sequences. Moreover, the alignment step is preceded by a filtering step which enables to only align couples of sequences that potentially have an identity greater than the chosen threshold. Finally to improve the performance of the package, the filtering and the alignment steps are parallelized according to the SIMD model.

3.1.5/ DACE

DACE is a parallel high performance clustering tool for very large data sets. It iteratively partitions the input data set into several non intersecting subsets before using the DP-means algorithm for clustering the subsets in parallel. Based on [Jiang et al., 2016],

DACE runs up to 80 times faster than its competitors including CD-HIT and UCLUST. However, this is only valid for very large data sets while for small data sets it might perform slower than the mentioned competitors.

3.1.6/ HPC-CLUST

HPC-CLUST is a tool featuring a distributed hierarchical clustering algorithm. It enables HPC-CLUST to cluster large data nucleotide sequences at high speed. In contrast with the previously described tools, HPC-CLUST takes aligned sequences as input, thus saving the computation time of the alignment task. The hierarchical algorithm starts by grouping close sequences, then it forms larger clusters by iteratively merging the closest groups.

3.2/ CLUSTERING POTENTIALLY DIVERGENT SEQUENCES

3.2.1/ GCLUST

In [Bruneau et al., 2018], a clustering Python module that uses Laplacian Eigenmaps and a Gaussian Mixture Model, was presented. Unlike most clustering packages, that largely utilize greedy approaches and just aim to improve the speed of clustering highly similar sequences, this one focuses mainly on improving the accuracy of the clustering for nucleotide sequences. Only few intelligent clustering tools use machine learning approaches. For instance, MeShClust [Girgis et al., 2018] is a multi-threading enabled package based on a mean shift algorithm. But based on its usage instructions, the input identity of the sequences parameter is the most important parameter. To our knowledge, the proposed package, namely GCLUST, is the first clustering method that uses unsupervised learning [Hastie et al., 2001] and does not require any sequences identity or centroid sequences user input. Such methods can help researchers make progress in a field where good balance between accuracy in clustering, even for potentially distant and divergent biological sequences, and high computational speed is required. Reaching this balance is unfortunately very difficult in practice. In particular, although the computational speed of the method proposed in [Bruneau et al., 2018] was demonstrated to be only moderately worse than the competitors on the ND3 test set of 100 sequences, its performance is significantly degraded when applied to larger sets composed of a few hundreds of sequences.

GCLUST is made of three main stages:

1. Sequences alignment using MUSCLE [Edgar, 2004]: in this phase, the input se-

quences are sent to the MUSCLE package in order to obtain an aligned set. MUSCLE is available as an external and independent executable module, in addition to its existence as a function in Python's COmparative GENomics Toolkit [cog,] (cogent). (See [Bruneau et al., 2018]for more details on how MUSCLE can be called from a Python package.)

2. Similarity matrix calculation: this phase relies on pairwise sequence comparison. For a set of N sequences, a NxN square matrix is computed. The value of the (i, j) element in this matrix is the similarity index between the pair of sequences i and j. This similarity index is based on the distance between these two sequences, which is calculated using the scoring matrix EDNAFULL.

3. Sequences clustering: this last stage is the core of the clustering method. It takes the similarity matrix as input, and clusters the sequences using the Laplacian Eigenmaps and the Gaussian Mixture Modelling.

The first and third stages depend on third party modules and existing libraries' functions, whereas the second stage, is an internally developed code to construct the similarity matrix. This latter is an intensive computation step with a complexity of order $O(\frac{N^2-N}{2})$, where N is the number of input sequences[1].

3.2.2/ AncestralClust

AncestralClust [Pipes et al., 2021] is a recently released tool that is developed for clustering divergent nucleotide sequences. It uses an iterative hierarchical method. Based on a recent research [Pipes et al., 2021], AncestralClust proceeds as follows: i- a user defined number of sequences are randomly selected from the input dataset, ii- a neighbor-joining phylogenetic tree is constructed for each one of the selected sequences (the number of sequences in this tree can be optionally defined by the user), iii- based on the branch lengths, the constructed trees are split into initial clusters, iv- the ancestral sequence of each initial cluster is inferred after aligning its sequences, v- the genetic distance is calculated between the remaining sequences and the ancestral sequences of the initial clusters, and based on this distance, each sequence is either assigned or not to one of these clusters. The algorithm iterates over these 5 steps until all the sequences are assigned to clusters.

[1] This matrix is symmetric and thus only the upper or lower diagonal must be computed.

3.3/ CONCLUSION

The description of the aforementioned tools shows that some of them use similar algorithms. For instance, the main difference between three of the tools that target highly similar sequences, is a minor change in the initial sorting process for the input sequences. These tools commonly implement a hierarchical algorithm except DACE. Moreover, a previous study [Chen et al., 2013] concluded that the estimation of the number of operational taxonomic units (OTUs)[2] is severely affected by the choice of the choice of the similarity threshold. This study compared the results of various clustering tools, including some of the ones presented above, in clustering 16S rRNA sequences.

Conversely, the tools that are designed to handle potentially divergent sequences do not require the choice of a similarity threshold, and implement two completely different algorithms. One of these tools hierarchically builds subsets of phylogenetic trees and detects the clusters based on the length of the branches, while the other tool suggests an interesting unsupervised ML algorithm that uses EM-GMM. The latter can also avoid the inconveniences of using phylogenetic trees where the phylogenetic signal can be scrambled in some cases, e.g. horizontal genes transfer. It also leaves enough room for improvements. The next chapter of this thesis presents various clustering validation indexes that will be useful later in assessing the quality of the clusterings obtained during our experiments.

[2]Clusters of closely related organisms grouped by DNA sequence similarity

4

CLUSTERING VALIDATION

In this chapter a few methods for evaluating the quality of a clustering are presented. These methods consist of calculating certain metrics that indicate if an obtained clustering is accurate or not. Based on [Schütze et al., 2008, clu,], a clustering can be evaluated either internally or externally. Although these two types of validation can be complementary in a qualitative study, they are not both always applicable; for instance, an external validation requires a previous knowledge of the correct clustering, conversely from the internal validation that do not have such a requirement.

4.1/ INTERNAL VALIDATION

An internal evaluation focuses on the intra-clusters and inter-clusters similarities, i.e., a good clustering should have a high intra-cluster similarity and a low inter-cluster one. These similarities are used in the computation of many internal clustering indexes [Wang et al., 2019, Guyeux et al., 2019, Somashekara et al., 2014, clu,] such as Silhouette, Davies-Bouldin, and Calinski Harabasz among others. A brief description about these indexes will be presented in the following subsections.

4.1.1/ SILHOUETTE

This index relies on two distances for each element i of the clustered dataset: $a(i)$ which is the mean distance between this element and the other elements belonging to the same cluster, and $b(i)$ which is the minimum distance between this element and all the other elements of the other clusters. If the size of i's cluster is equal to 1, the Silhouette value $s(i)$ for this element i is equal to 0, otherwise, this value is defined by the following formula:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}.$$

The Silhouette index for a given clustering is the average of the Silhouette indexes of all its elements. This index ranges between -1 (indicating an inappropriate clustering) and 1 (for an appropriate one).

4.1.2/ DAVIES-BOULDIN

This index is equal to the mean similarity between each cluster C_i and its most similar one C_j. The similarity $R_{i,j}$ between two clusters i and j is the sum of their diameters divided by the distance between their centroids. Let k be the number of clusters in a given clustering. The Davies-Bouldin index is calculated as follows:

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_{i\neq j} R_{i,j}.$$

The DB index is a positive decimal. A small value for this index indicates that the clusters are well separated and thus, the quality of the clustering is good.

4.1.3/ CALINSKY-HARABASZ

Also known as the variance ratio criterion, this index relies on two variances: the overall within-Cluster variance SS_W and the overall between cluster variance SS_B [cal,]. Let N be the number of the clustered elements (observations) and k the number of clusters in the obtained clustering. The Calinski-Harabasz index is calculated as follows:

$$CH = \frac{SS_B}{SS_W} \times \frac{N-k}{k-1}.$$

The CH index is a positive decimal. Contrary to the DB index, a higher value of the CH index indicates a better clustering.

4.2/ EXTERNAL VALIDATION

The external validation indexes are also used for the quality assessment of a certain clustering, in the cases where they are applicable. Conversely from the internal clustering validation indexes, the external validation indexes require both the true[1] and the resulting clusterings. These indexes provide a better assessment of a resulting clustering since they compare it to a human defined truth. Two major aspects are assessed when computing the external indexes:

[1] also called gold standard or golden truth

- The existence in a resulting cluster items that should be in another cluster.
- Obtaining a correct number of resulting clusters.

4.2.1/ PURITY

The *purity* metric can either be computed for each cluster individually or for the whole clustering. The *purity* of a cluster [Schulte im Walde, 2003] can be computed as defined in equation 4.1. The purity of a given cluster has a value in the interval $]0, 1]$. It will be close to zero if the cluster's elements are found in different clusters in the perfect clustering and on the other hand it will increase up to 1, if the cluster have all its elements from the same cluster in the perfect clustering. The quality of a single cluster C_i is computed according to the maximum of p^i_j which is the number of elements that C_i has in common with the cluster j in the perfect clustering.

$$purity(C_i) = \frac{1}{|C_i|} * max_j(p^i_j) \tag{4.1}$$

In addition, as defined in [Schütze et al., 2008] and [clu,], for a dataset of N elements, the *purity* of a certain clustering K, formed by n clusters k_i $(1 \leq i \leq n)$, with regards to a reference clustering C, formed by m clusters c_j $(1 \leq j \leq m)$, is calculated as follows:

$$purity(K) = \frac{1}{N} * \sum_{i=1}^{n} max^m_{j=1} |k_i \cap c_j| \tag{4.2}$$

The *purity* of a clustering ranges from a value close to 0, for randomly mixed items, to 1 for a pure clustering. Although it is able to penalize the existence of items that are wrongly classified together, this index presents two major drawbacks.

1. It is not suitable for imbalanced clusters. For example, the worst *purity* will score 0.970 in a case where a true clustering consists of 2 clusters, with the first cluster containing 100 elements and the second cluster containing only 3 elements.
2. It does not penalize the division of a certain cluster into sub-clusters. Indeed, if all the items are clustered as singletons, then a perfect *purity* score of 1 is obtained.

4.2.2/ ADJUSTED RAND INDEX

Beside the *purity*, the Adjusted Rand Index (ARI) is another external validation index that is frequently used [Santos et al., 2009]. This index tests the clustered items by pairs and checks if the items in this pair are correctly clustered together or apart. Contrary to the purity index, a wrong resulting number of clusters is also penalized in this method.

Therefore, the ARI overcomes the limitations of the $purity$ and better reflects the quality of the clustering. This index ranges between 0 for two completely different clusterings to 1 for identical clusterings. The compared clusterings do not have to have the same number of clusters. Moreover, the ARI is symmetric: swapping or changing the clusters' labels, in a certain set, does not affect the calculation, e.g. the sets [a,a,a,b,b,c,c] and [c,c,c,a,a,d,d] are still detected identical although the labelling of the clusters' elements was changed. But the ARI and the $Purity$ indexes are complementary and should be computed together to determine if a non-perfect ARI score is related to impurities or wrong number of resulting clusters or both. $Purity$ and ARI were both used to assess the results of the experiments presented in the subsequent chapters.

4.3/ CONCLUSION

The internal and the external validation indexes both play a crucial role in the analysis of the clustering quality. The internal validation indexes can be computed for any resulting clustering, without the need of any additional knowledge about the clustered data. Conversely, although they require an a priori knowledge of the true clustering, the use of the external validation indexes remains unavoidable because a good score in an internal validation index does not necessarily infer a good quality clustering in a real application as shown in [Schütze et al., 2008]. Therefore, the choice of using either one or both of the validation indexes types, depends on the available data for their computation. The presented validation indexes will be used to assess the quality of the produced clusterings. In the next chapters, the main contributions of this thesis are presented.

III

CONTRIBUTIONS

This part examines the main contributions in this thesis. As previously discussed, the proposed approaches and methods are designed to solve different challenges, starting from assuring a reliable clustering for potentially divergent biological sequences. It extends to improve the performance and accuracy of the GMM-based tools for clustering biological sequences, in addition to validating and proposing additional and novel algorithms for this clustering purpose.

5

CLUSTERING POTENTIALLY DIVERGENT SEQUENCES

5.1/ INTRODUCTION

Mutations and substitutions, in the nucleotide sequences, happens in different rates and for many reasons [Duffy et al., 2008, Wielgoss et al., 2013, Oliver et al., 2010]. While some substitutions and mutations happen as a natural environment adaptation process, e.g. the crisis-causing bacterial adaptation to antibiotics [Gullberg et al., 2011, Ventola, 2015] that is emerging as a reason of excess of mortality [Lim et al., 2016], others might be linked to some artifacts, such as the exposure to some pollutants, and result in diseases and anomalies [Sørensen et al., 2003], e.g. cancerous cells.

As a result of the increasing number of mutations' causes and the large number of new sequences discoveries, linking these sequences to their siblings and ancestors becomes more complex and the use of clustering tools is essential to tackle this problem. Many clustering tools, based on hierarchical or greedy algorithms, relying on a user input similarity threshold, and targeting high speed clustering of highly similar sequences, currently exist and some of them became widely used. Some of these tools use parallel computing to provide even higher clustering speeds. In [Bruneau et al., 2018], an innovative clustering module for genomic sequences that uses Laplacian Eigenmaps and a Gaussian Mixture Model, was presented. The first implementation of the algorithm gave very promising results when compared to other existing tools, especially in terms of clustering accuracy of potentially divergent sequences. However, since it computes the similarity matrix between all the input sequences which is a computationally intensive operation, its execution time significantly increases when clustering large sets of input sequences. Therefore, reaching a good speed and accuracy in nucleotide or protein sequences clustering, involving large numbers of divergent sequences, is a very challenging problem.

In the present chapter, a new implementation of the clustering algorithm is presented. A

special attention was given to improve the overall performance and scalability of this new version. The new hybrid C++/Python clustering package, called SpCLUST, computes in parallel the similarity matrix using the Message Passing Interface (MPI) which drastically reduces the execution time of this stage. This new package was integrated into a GALAXY platform [Afgan et al., 2018] and is freely available online. Many experiments were conducted to evaluate the accuracy and the performance of SpCLUST while using simulated and real data sets. It was also compared to other clustering packages, such as UCLUST, CD-HIT and DNACLUST.

The rest of this chapter is organized as follows. In Section 5.2, the improvements added to the initial Python package are detailed. In Section 5.3, a performance comparison between the different versions of the package is presented. It is followed by a comparative study between SpCLUST and other packages. The chapter ends with a discussion that recapitulates the presented contributions.

5.2/ SpCLUST: AN IMPROVED CLUSTERING PACKAGE

In this section, we recall the main ideas underlying the construction of the Python module in [Bruneau et al., 2018] and describe the main proposed changes for improving its performance and expanding its functionality. We also present a description of each submodule of the code.

5.2.1/ ANALYSIS OF THE ORIGINAL PYTHON PACKAGE

The main objective of the original package was to provide a good clustering method. For a given relatively divergent sets of sequences and without a previous knowledge of the number of clusters, it should produce high quality clusters with high intra-class similarity and low inter-class similarity. It solely focused on the quality of the clusters and little measures were taken to improve the performance of the method. Therefore, in this chapter, great emphasis was placed on improving the performance of this clustering method. A thorough analysis of the execution time of each stage of the package was performed in order to detect the main bottlenecks.

This analysis was done using two data sets consisting of 100 and 1024 sequences respectively, and using two computers equipped with different processors: an i3-5005U 2.0GHz dual-core (4 core threads) processor and an i7-6700 3.4GHz quad-core (8 core threads) processor. The profiling results are displayed in Table 5.1.

As shown in Table 5.1, the similarity matrix calculation, which is mostly composed of the distance matrix computation, is by far the most time consuming task in the overall

	i3-5005U 2.0GHz processor		i7-6700 3.4GHz processor	
	100 sequences	1049 sequences	100 sequences	1049 sequences
Alignment phase	8 seconds	73 minutes	5 seconds	36 minutes
Similarity matrix calculation phase	18 minutes	5696 minutes	9 minutes	2848 minutes
Clustering phase	7 seconds	33 minutes	4 seconds	15 minutes
Total	18 minutes	5802 minutes	9 minutes	2899 minutes

Table 5.1: Execution time of the original Python package.

pipeline. Therefore, optimizing the computation of the similarity matrix should drastically reduce the overall execution time of the clustering package.

In consequence, most of the modifications presented in this chapter concern this stage. In particular, it was re-implemented in the C++ programming language which, although it is a more complex programming language than Python [Oliphant, 2007], is clearly faster [Fourment et al., 2008]. Moreover, to furthermore reduce the execution time of the computation of the similarity matrix which computes independently the similarity indexes between all the sequences, it was parallelized using MPI according to the Master/Slave model. The content of each phase of the package and the added modifications to each one of them are detailed in the next subsections

5.2.2/ ALIGNMENT PHASE

This phase consists of obtaining an aligned version of the input sequences. Many alignment packages, such as MUSCLE [Edgar, 2004], T-Coffee [Notredame et al., 2000], MAFFT [Katoh et al., 2013], PASTA [Mirarab et al., 2015], and ClustalW [Thompson et al., 2002], are available but have different accuracy and performance. Notredame states in [Notredame et al., 2000] that T-Coffee provides a dramatic improvement in accuracy with a modest sacrifice in speed. On the other hand, ClustalW is widely used, has cross-platform releases and offers both command line and a graphical user interface version. Kuo-Bin Li also worked on improving its performance by introducing in [Li, 2003] a parallel version, called ClustalW-MPI.

However, MUSCLE remains better supported than ClustalW for command line calls, requiring no user intervention, and can on average achieve both higher accuracy and lower execution time than ClustalW or T-Coffee, depending on the chosen options [MUS,]. For instance, in the case of aligning large data sets which is an extremely time consuming task, after just two iterations, MUSCLE gives an alignment with a precision equal to the one computed by T-Coffee and takes less time than ClustalW [MUS,]. Moreover a study, published in [Deng et al., 2006], proposes a parallel computation version of MUSCLE that should theoretically improve further its performance. However, the implementation of the proposed parallel version of MUSCLE was not found online.

Further benchmarks [Wilm et al., 2006, Ahola et al., 2006, Nuin et al., 2006] showed that MAFFT outperforms the other tools, including MUSCLE, in terms of alignment's speed and quality. Moreover, MAFTT has a multi-threaded version where the alignment can be computed in parallel using all the available core threads in a workstation. However, unlike the results presented in [Wilm et al., 2006, Ahola et al., 2006, Nuin et al., 2006], our alignment test, performed on a set of random nucleotides sequences and conducted over a workstation equipped with a dual-core (4 core threads) 2.0GHz processor, gave the following results: MUSCLE's official and sequential module computed the alignment in 11 seconds while the multi-threaded version of MAFFT took 13 seconds and the sequential one required 18 seconds. Moreover, MUSCLE had the following advantages:

- It is a single 4MB executable file, whereas MAFFT's module is a package containing over 100 files with a total size higher than 60MB after extraction.
- It does not generate any interfering outputs when called with the "-quiet" parameter, unlike MAFFT that only omits the alignment progress output using this parameter.

For these reasons, the proposed SpCLUST package continues on using MUSCLE for aligning the sequences.

5.2.3/ SIMILARITY MATRIX COMPUTATION PHASE

In the original package, for each pair of aligned sequences, the distance between them is computed using the EDNAFULL [EDN,] scoring matrix. They are stored in the distance matrix, where the element of index (i, j) contains the distance between the i^{th} and j^{th} sequences. The similarity matrix is then computed from the distance matrix. In the new version, this procedure was re-coded in C++ to reduce its execution time and it was extended to use two additional scoring matrices: BLOSUM62 and PAM250 [Sub,] which can be specified as a parameter. The added scoring matrices extends SpCLUST's operational scope to include protein sequences clustering.

Since the computation of the distance matrix is the most time consuming phase of the clustering package and it is quadratically proportional to the size of the input sequences, it was also parallelized to reduce its execution time. The parallel version uses MPI to distribute the computations. For each available core thread on the used workstation, a slave process is created and the master process assigns to each slave process an equal number of sequences pairs. The distances between the assigned pairs of sequences are computed in parallel on P slave processes and sent back to the master which stores them in the distance matrix. This inter-process interactions are illustrated in Figure 5.1. Since the computed distance between two sequences is a commutative operation, the resulting

distance matrix is symmetric. Therefore, only the upper triangular matrix is computed and the lower one is the transpose of the upper triangular matrix, such as $d_{(i,j)} = d_{(j,i)}$ for $j < i$.

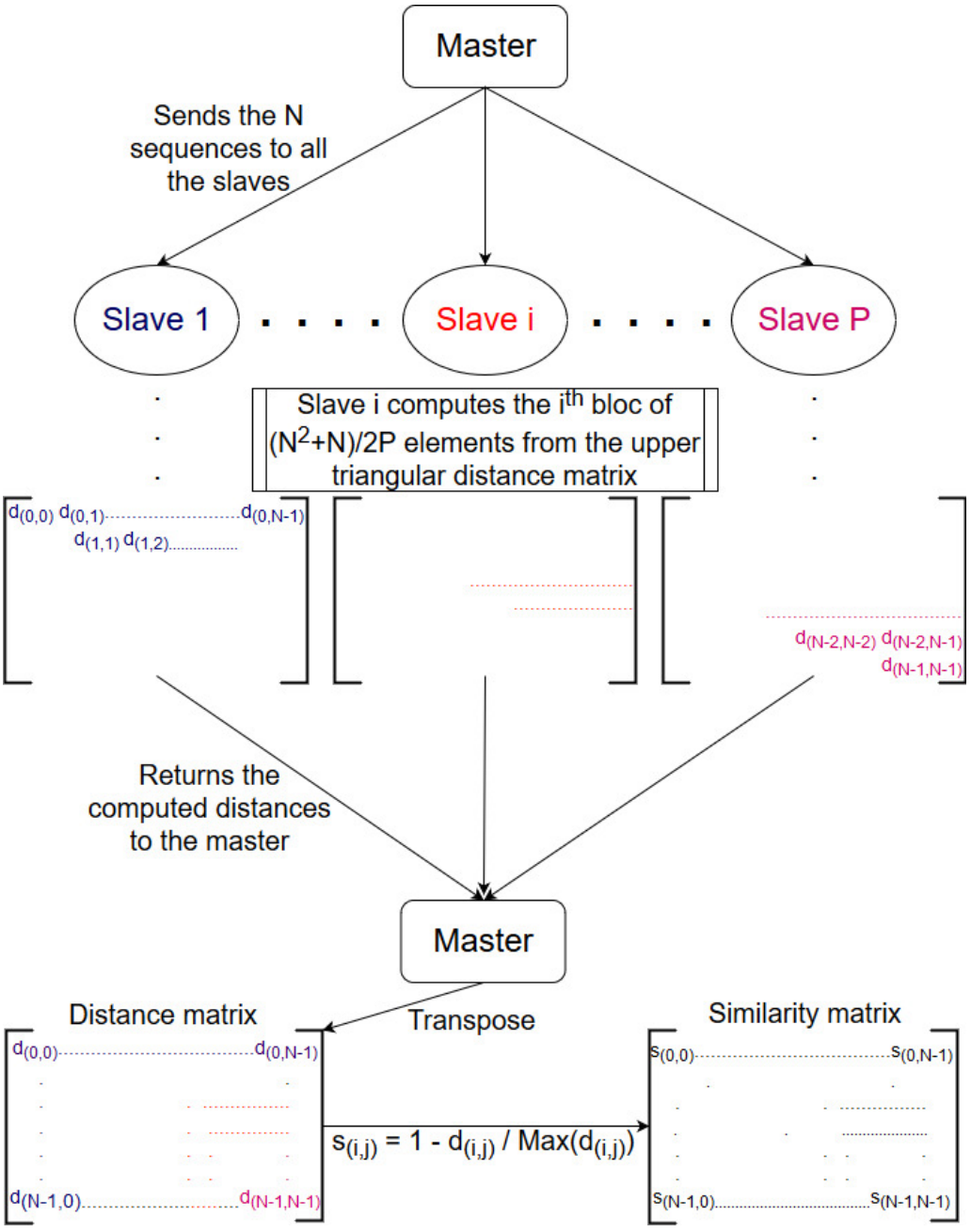

Figure 5.1: Processes Master-Slave architecture.

5.2.4/ CLUSTERING PHASE

This phase uses the previously calculated similarity matrix to cluster the sequences. It also relies on the use of the Laplacian Eigenmaps and the Gaussian Mixture Model, as a machine learning model, in order to produce the clustering. In the new version,

obsolete functions were replaced and some parameter calibration were necessary to get the same clusters as in [Bruneau et al., 2018], for 100 DNA sequences taken from the mitochondrially encoded NADH dehydrogenase 3 (ND3) gene.

5.2.5/ PACKAGE AVAILABILITY

SpCLUST is available in command line versions for both Windows and Linux. A basic graphical user interface allows the user to browse input and output files, and to view the obtained clustering. The source code and the executable files of the SpCLUST package are available online[1]. They could be customized by passing parameters that can modify the quality and the execution time of the alignment performed by MUSCLE. The latest version is also integrated to a publicly accessible GALAXY server[2]. It can be found under the menu item “SpCLUST”.

5.3/ PERFORMANCE EVALUATION OF SPCLUST

After describing the workflow and the improvements offered in the new clustering module, the performance of the different versions will be investigated. The following experiments will detail the performance of each phase in the proposed module.

5.3.1/ ALIGNMENT PHASE

In this section, the execution time of the alignment phase, performed by MUSCLE, is presented for different sizes of data sets, different number of iterations, and while running on different workstations. Figures 5.2 and 5.3 show the time taken to align, in different numbers of iterations, two data sets using the 2.0GHz i3-5005U processor and the 3.4GHz i7-6700 processor.

For large data sets, the choice of the number of iterations drastically impacts the alignment time. Therefore, a trade-off between the quality of the alignment and its execution time should be considered. The impact of this choice on the clustering quality will be discussed in Section 5.4.2.3.

5.3.2/ SIMILARITY MATRIX CALCULATION PHASE

As mentioned in Section 5.2, the similarity matrix calculation was re-implemented in C++ then parallelized using MPI to reduce its execution time. Figure 5.4 illustrates the execu-

[1] https://github.com/johnymatar/SpCLUST
[2] http://galaxy.ul.edu.lb

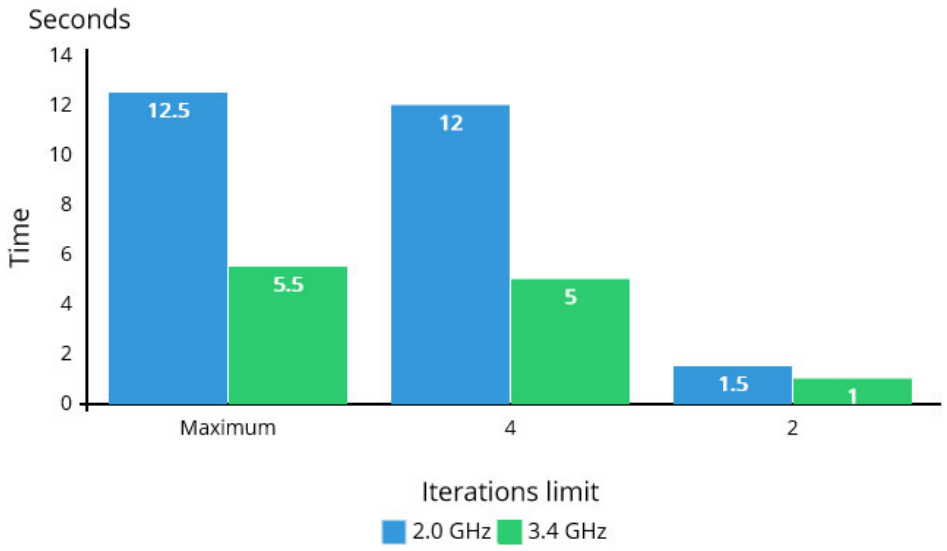

Figure 5.2: Alignment time for the 100-sequences ND3 set.

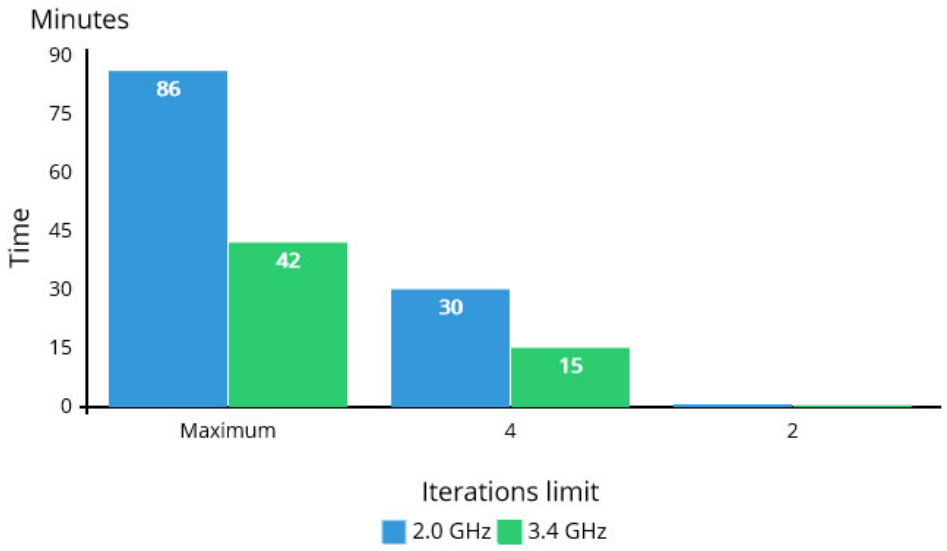

Figure 5.3: Alignment time for the 1049-sequences set.

tion times taken by each version to compute the similarity matrix of the 1049-sequences set with the three scoring matrices. These experiments were conducted on a workstation equipped with the I3-5005U 2.0GHz dual-core Intel processor. The parallel version was using the four core threads of that processor.

The results in Table 5.1 show a huge performance improvement when compared to the execution time needed by the initial Python module. The modulated version is a C++ version where the main module and the sequence pairwise distance calculation module were two separate executables. It took 342 minutes to compute the similarity matrix using the EDNAFULL scoring matrix while the original Python module required 5696

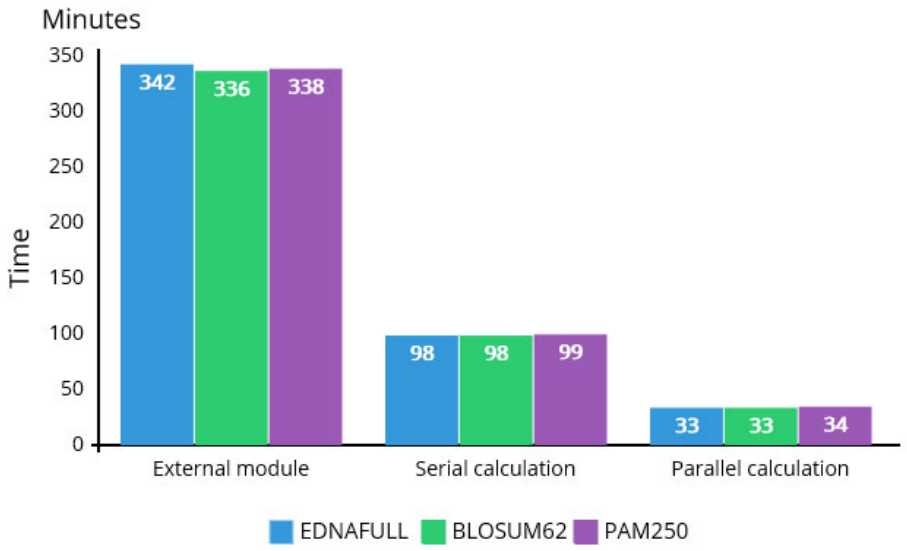

Figure 5.4: Similarity matrix calculation time using the I3-5005U 2.0GHz dual-core (4 core threads) processor.

minutes, resulting in around 16.6X speed-up. The serial one executable version took just 99 minutes, showing a speed-up of 57.5X. Finally, the parallel version, running on 4 core threads, took 34 minutes, a 167.5X speed-up over the Python package and a 2.91X speed-up over the serial C++ version.

The results also show that for all the versions, the similarity matrix calculation time is nearly the same, regardless of the chosen scoring matrix: the use of the newly integrated matrices does not affect the performance of the package.

Moreover, to study the scalability of the parallel similarity matrix computation, its normalized execution time and its strong scaling efficiency[3] were measured. Figure 5.5 shows the normalized execution time and the strong scaling efficiency of this matrix computation using different number of slave processes running on the 3.4GHz I7-6700 quad-core (8 core threads) processor for the same data set. As shown in Figure 5.5, the strong scaling efficiencies with 2, 4 and 8 slave processes were equal to 96%, 80% and 59% respectively. Those numbers demonstrate that although the parallel version do not scale linearly due to communication and I/O overheads, its execution time is continuously reduced while increasing the number of used slave processes up to the number of available core threads.

A similar experiment was conducted on a workstation equipped with 4 slower 1.87GHz quad-core XEON E7520 processors and using up to 16 core threads. Figure 5.6 shows the normalized execution time and the strong scaling efficiency of the parallel similarity

[3] If the amount of time to complete a work unit with 1 processing element is $t1$, and the amount of time to complete the same unit of work with N processing elements is tN, the normalized execution time is $(tN/t1) * 100$ and the strong scaling efficiency is $t1/(N * tN) * 100$

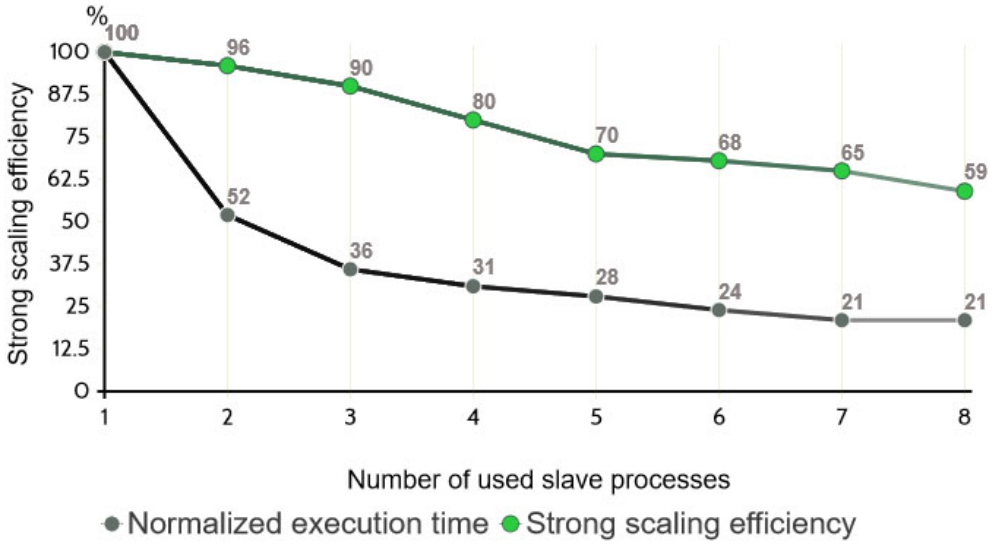

Figure 5.5: Scalability on a 3.4GHz processor with 8 core threads.

matrix computation. As in the previous experiment, and as shown in Figure 5.6, it can be noticed that the strong scaling efficiency of this parallel application slowly decreases from 99% with 8 slave processes to 59% with 16. It can also be noticed that the optimal number of slave processes to use is 8, and beyond this number the IO overhead from reading sequences in parallel significantly reduces, on this workstation, the scalability of the parallel similarity matrix computation. This second workstation is currently hosting the publicly accessible GALAXY server with the latest SpCLUST version.

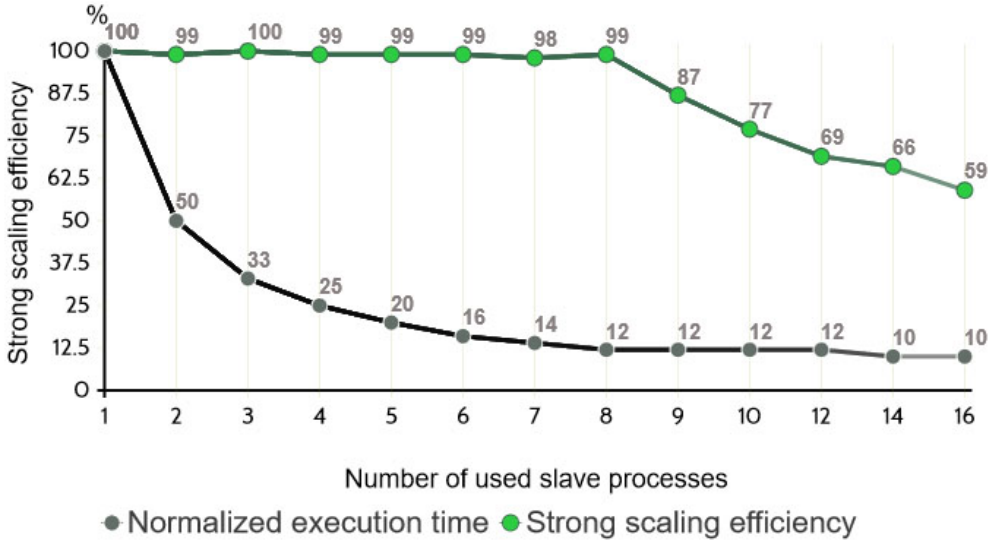

Figure 5.6: Scalability on four 1.87GHz processors using 16 core threads.

To conclude the scalability study, the similarity matrix computation time was recorded for

five larger sets of sequences. These sets contained 2000, 5000, 10000, 20000 or 30000 sequences, extracted form a set of aligned archaea nucleotide sequences that was retrieved from the Linux package of HPC-CLUST[4]. These experiments were performed over a cluster of 34 nodes, where each node has a 3.4GHz processor with 4 cores and 2GB RAM. Figure 5.7 shows the recorded computation time for each input number of sequences, ranging from 0.66 minutes for the 2000-sequence set to 42 minutes for the 30000-sequence set. As previously shown, the complexity of the similarity matrix computation is of order $O(\frac{N^2-N}{2})$, where N is the number of input sequences. Therefore, the number of operations should scale quadratically to N. However, the experiments show that the computation time of the similarity matrix scales on a slower rate. For example, between the 5k and the 10k sets the computation time scales by a ratio of 3 while the number of operations was multiplied by 4. Similarly, between the 10k and the 20k sets, the computation time only scales by a ratio of 3.5. These ratios show that the proposed solution is indeed scalable and the difference of proportionality comes from communication time needed to distribute the increasing data set to the cluster's slave processes over the network.

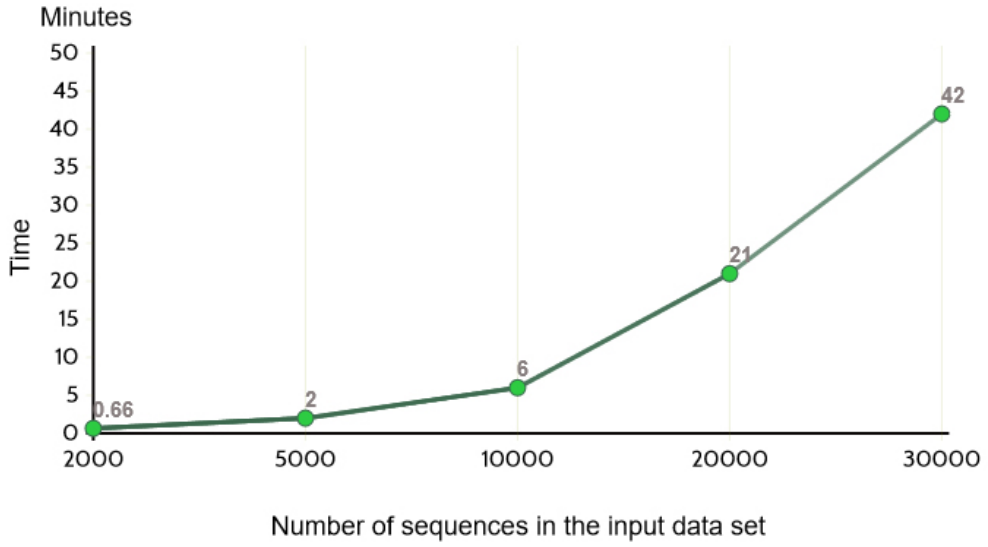

Figure 5.7: Scalability on a cluster of 34 nodes having 3.4GHz processors.

5.3.3/ CLUSTERING PHASE AND OVERALL PERFORMANCE

The clustering phase's sub-module, as stated in Section 5.2, was updated and uses the latest Gaussian Mixture model package [Gau, a]. The recorded runtime values reflect the clustering phase's performance after the replacement of the obsolete functions. These

[4]https://www.meringlab.org/software/hpc-clust/hpc-clust-1.2.1-bin.tar.gz

tests allow us to conclude the profiling analysis and assess the overall performance improvement of the full package. Figure 5.8 shows the detailed profiling results of SpCLUST while running on the 2.0GHz dual-core (4 core threads) i3-5005U processor and for a set of 1049 sequences. It took 153 minutes while using a maximum precision alignment and just 67 minutes with a fast alignment. These values show a speed-up of 37.9X and 86.5X, respectively, when compared to the original Python module. Moreover, the total runtime, on the quad-core (8 core threads) i7-6700 3.4GHz and for the same set of 1049 sequences, was equal to 65 minutes using a maximum precision alignment and 23 minutes using a fast alignment, giving a 44.6X and 126X speed-up, respectively.

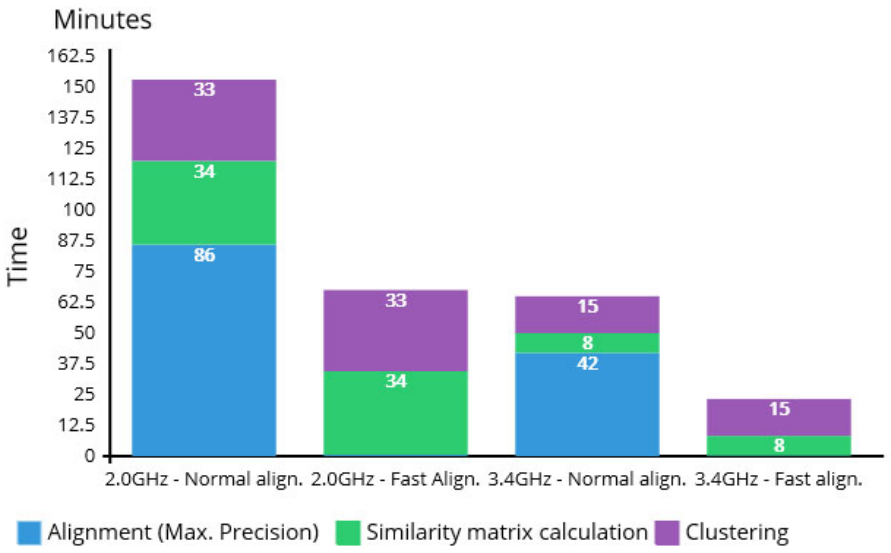

Figure 5.8: Run time for all phases execution.

5.4/ A COMPARATIVE STUDY BETWEEN SPCLUST AND COMPETING TOOLS

This section presents a comparative study between SpCLUST and four competing clustering tools. The experimental protocol is first described, then the clustering results are compared in terms of number of clusters and their contents. Finally, the effect of the alignment quality on the clustering is discussed.

5.4.1/ EXPERIMENTAL PROTOCOL

The comparative study, interpreted in this section, used eighteen sets of simulated data (12 genomic sets and 6 protein sets) and eight sets of real data (4 genomic sets and 4 protein ones). The simulated data sets derive from the following reference sequences:

- four different NADH dehydrogenase 3 (ND3) sequences, extracted from a collection of Platyhelminthes and Nematoda species. The mutated sequences, generated from these four reference sequences, should produce four clusters.
- five different protein sequences (should produce five clusters).
- six different gene sequences extracted from chloroplastic genomes (should produce six clusters).

From each group of reference sequences, new mutated sets which contains 30 mutations from each initial sequence, were generated. Each set had different properties in terms of mutation criteria and divergence rate between its sequences. The mutated sets were divided into two categories based on the used transition. The transition is performed in either of the following two ways:

1. the nucleotide or protein transition is performed on a random base.
2. the nucleotide transition is performed according to a real computed rate for URA3, published in [Lang et al., 2008], whereas the protein transition is performed according to the rates of the PAM1 substitution matrix.

Each category includes three mutated sets from each group of reference sequences, namely $\{$S1, S2, S3$\}$ and $\{$S'1, S'2, S'3$\}$ for each group. The mutation is performed using the following criteria:

- the S1 and S'1 are the result of four generations of mutation with 15% mutation rate, 10 maximum random insertions of size inferior to 9 nucleotides or proteins and a gap rate equal to 30% of the number of insertions with a maximum gap size of 10.
- the S2 and S'2 are the result of four generations of mutation with 10% mutation rate, 7 maximum random insertions of size inferior to 6 nucleotides or proteins, and a gap rate equal to 20% of the number of insertions with a maximum gap size of 7.
- the S3 and S'3 are the result of two generations of mutation with 5% mutation rate, 4 maximum random insertions of size inferior to 4 nucleotides or proteins, and a gap rate equal to 10% of the number of insertions with a maximum gap size of 4.

Based on the mentioned criteria, the sets S1 and S'1 contain the most divergent sequences when compared to the initial ones, whereas S3 and S'3 contain the least divergent ones. Figure 5.9 shows part of a ND3 sequence on which four generations of simulated mutations were performed. The first row contains the original sequence content and the next four rows show the added mutations, gaps and insertions from one generation to the other.

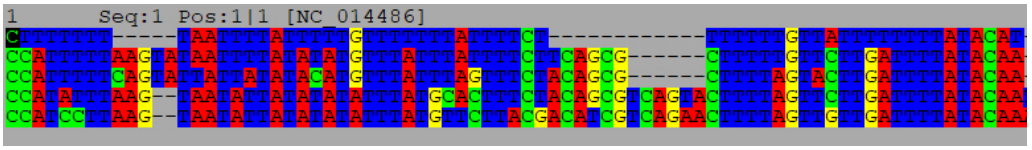

Figure 5.9: ND3 simulated mutation

The real data sets were formed from mixes of genomic or protein sequences gathered and downloaded using NCBI's HomoloGene online tool[5] and NCBI's virus resources[6,7]. The starting genomic or protein sequences are from:

- homologous genes to the human genes FCER1G, S100A1, S100A6, S100A8, S100A12 and SH3BGRL3 which all belong to the first chromosome, and to the human gene MC1R which belongs to the 16^{th} chromosome. These homologous genes are extracted from a collection of mammal species.
- variants for segments PB2 and PB1 from the most fatal type A influenza's serotypes [Inf,]. These serotypes are H1N1, H2N2, H3N2, H5N1, and H7N9.
- variants for the C-E genome region of the Zika virus.

The sequences in each data set were randomly shuffled. The following describes the content of each data set and how many clusters are expected in the clustering results:

- Set 1 consists of six series of homologous genes to the FCER1G, S100A1, S100A6, S100A8, S100A12, and SH3BGRL3 genes which belong to the human's first chromosome. The sequences should be separated into six clusters.
- Set 2 contains a series of homologous genes to the MC1R gene, found in the human's sixteenth chromosome, in addition to the series in the first set. The clustering of set 2 should therefore produce seven clusters.
- Set 3 contains variants of the segment PB2 of the influenza type A serotype and should produce five clusters.
- Set 4 contains variants of the Zika's C-E segment, influenza's AH1N1 PB2 segment, and influenza's AH2N2 PB1 segment and should produce three clusters.

All these data sets are available on SpCLUST's GitHub repository[8].

In this comparative study, besides SpCLUST, six other clustering tools were considered: CD-HIT, UCLUST, DNACLUST, SUMACLUST, DACE, and HPC-CLUST. However, since

[5] https://www.ncbi.nlm.nih.gov/homologene
[6] https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi#mainform
[7] https://www.ncbi.nlm.nih.gov/genomes/VirusVariation/Database/nph-select.cgi
[8] https://github.com/johnymatar/SpCLUST

SUMACLUST failed to run correctly on many data sets and HPC-CLUST generated clusters containing only singletons on most of the data sets, only the remaining four tools were evaluated and compared to SpCLUST.

To cover multiple levels of sequence divergence, we used 3 similarity thresholds: 0.9, 0.6, and 0.4. These values have been chosen according to [Pearson, 2013], an article investigating homologous sequences similarity for a wide range of organisms. According to this research work, 0.9 identity allows to identify highly similar groups of sequences, while identities ranging from 0.4 to 0.6 cover the majority of highly to moderately divergent groups of sequences.

5.4.2/ Results comparison and interpretation

The interpretation of the experiment's results is organized in three tracks. The number of clusters, in each clustering, is discussed in the first track, the clusters' contents are assessed in the second one, whereas SpCLUST's results, using a high quality (slow) or fast alignment, are compared in the third track.

5.4.2.1/ Analysis of the number of returned clusters

The number of clusters in the clustering results of each evaluated tool, reflects a first aspect of the clustering accuracy. Obtaining a number of clusters equal or relatively very close to the real value, is an essential but not a sufficient condition for considering it to be a good quality clustering. The content of the clusters must corresponds to the reality too.

The data presented in Table 5.2 show the resulting clusters' numbers for the simulated sets that were produced from the sequences previously mentioned in the experimental protocol. In Table 5.2, "Prot." and "Clp" are the abbreviations of "Protein" and "Chloroplast". On the other hand, an "E" value indicates that the concerned tool was not able to cluster the specified data set with the given options, e.g. DNACLUST and DACE failed to handle the large chloroplast sequence sets while CD-HIT-Est failed to run with similarity thresholds equal to 0.6 or 0.4. Moreover, a "-" indicates that the current tool or parameter is not designed to handle this type of sequences, i.e. this tool is designed for genomic sequences while the set contains protein sequences, or vice versa. Finally, two tools of CD-HIT were used: for the genomic sets CD-HIT-Est was used whereas for the protein sets it was replaced by CD-HIT-Protein.

As described in the experimental protocol, the expected number of clusters for each data set are:

- 4 clusters for the ND3 simulated sets.

	CD-HIT			DNACLUST			UCLUST			DACE		SpCLUST		
Similarity threshold or scoring matrix	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	0.6	0.4	DNAF.	PAM.	BLOS.
ND3 simul. set S1	124	E	E	124	36	1	124	30	13	E	E	2	-	-
ND3 simul. set S2	94	E	E	110	23	1	106	18	10	E	E	5	-	-
ND3 simul. set S3	76	E	E	99	4	1	65	4	3	8	8	4	-	-
ND3 simul. set S'1	124	E	E	124	36	1	124	31	20	E	E	2	-	-
ND3 simul. set S'2	85	E	E	109	18	1	110	14	13	E	E	4	-	-
ND3 simul. set S'3	68	E	E	97	4	1	74	5	4	22	22	5	-	-
Prot. simul. set S1	155	54	26	-	-	-	155	66	40	E	E	-	4	5
Prot. simul. set S2	110	36	13	-	-	-	151	46	17	E	E	-	6	6
Prot. simul. set S3	89	5	4	-	-	-	114	8	4	40	22	-	6	6
Prot. simul. set S'1	155	58	34	-	-	-	155	51	34	E	E	-	5	5
Prot. simul. set S'2	113	34	15	-	-	-	150	36	13	E	18	-	6	5
Prot. simul. set S'3	102	5	4	-	-	-	112	7	4	21	E	-	5	5
Clp. simul. set S1	186	E	E	E	E	E	186	63	32	E	E	6	-	-
Clp. simul. set S2	133	E	E	E	E	E	142	40	24	E	E	5	-	-
Clp. simul. set S3	78	E	E	E	E	E	86	6	6	E	E	4	-	-
Clp. simul. set S'1	186	E	E	E	E	E	186	65	43	E	E	5	-	-
Clp. simul. set S'2	133	E	E	E	E	E	146	45	33	E	E	5	-	-
Clp. simul. set S'3	72	E	E	E	E	E	89	6	6	E	E	5	-	-

Table 5.2: The number of clusters returned by each clustering tool for the simulated sets.

- 5 clusters for the protein simulated sets.
- 6 clusters for the simulated sets derived from chloroplast genes.

A general overview of Table 5.2 shows that in the cases of the highly and moderately divergent sets, S1, S'1, S2 and S'2, all the tools except SpCLUST returned a number of clusters far from what was expected. Conversely, for the least divergent sets, S3 and S'3, these same tools, except DACE, gave the exact number of clusters or a close number to the expected one. In particular, it can be noticed that DNACLUST, while using a similarity threshold equal to 0.6, returned the expected number of clusters for the ND3 simulated sets, S3 and S'3. Moreover, UCLUST, while using a similarity threshold equal to either 0.6 or 0.4, produced the expected number of clusters for the ND3 and chloroplast genes simulated sets, S3 and S'3. In addition, CD-HIT-Protein, while using a similarity threshold equal to 0.6, was able to find the expected number of clusters for the protein simulated sets, S3 and S'3. Finally, SpCLUST gave either the exact number of clusters or a close one for all the simulated sets, including the most divergent ones.

The above observation shows that CD-HIT, DNACLUST and UCLUST are efficient in determining a correct or closely correct number of clusters in the case where the clusters' member sequences are convergent. For instance, for the least divergent simulated sets, these tools were efficient in most cases when applied with a similarity threshold equal to 0.6 or 0.4. But, unlike SpCLUST, these tools failed to cluster the relatively divergent sets.

Table 5.3 shows the numbers of clusters, produced by each tool while clustering the real

data sets. The expected numbers of clusters are equal to 6 clusters for the first set, 7 for the second set, 5 for the third set and 3 for the fourth set. CD-HIT-Est, and similarly to the experiments performed on the simulated sets, failed to run using the similarity thresholds 0.6 or 0.4. CD-HIT-Protein also failed while processing the protein sequences of the third set. DNACLUST failed to process the genomic sequences of the first and second sets, while DACE failed to process all the genomic sets. DACE also failed processing three out of the four protein sets using the similarity thresholds 0.9 or 0.6.

It is important to highlight that both High Performance Computing tools, DACE and HPC-CLUST, were successfully tested using a data set provided by one of their authors and containing tens of thousands of sequences. The main difference between this data set and ours is the length of the sequences. Therefore, although these tools might be well optimized to cluster large number of sequences, it seems they cannot handle lengthy sequences which are very common in real life cases.

	CD-HIT			DNACLUST			UCLUST			DACE			SpCLUST		
Similarity threshold or scoring matrix	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.6	0.4	DNAF.	PAM.	BLOS.
Set 1 genomic	33	E	E	E	E	E	33	17	8	E	E	E	4	-	-
Set 2 genomic	40	E	E	E	E	E	40	19	6	E	E	E	4	-	-
Set 3 genomic	5	E	E	10	43	44	5	1	1	E	E	E	5	-	-
Set 4 genomic	3	E	E	10	10	24	3	3	1	E	E	E	4	-	-
Set 1 protein	19	7	6	-	-	-	19	7	6	E	E	5	-	4	4
Set 2 protein	25	9	7	-	-	-	25	9	7	E	E	5	-	3	4
Set 3 protein	1	E	E	-	-	-	1	1	1	3	1	1	-	3	4
Set 4 protein	3	3	3	-	-	-	3	3	3	E	E	1	-	4	4

Table 5.3: The number of clusters returned by each clustering tool for the real data sets.

The results displayed in Table 5.3 show that, for the genomic sets, SpCLUST succeeded in producing the exact number of clusters for the third set and a close number for the others. CD-HIT-Est and UCLUST produced the exact number for the third and fourth sets while using a similarity threshold equal to 0.9, and UCLUST returned close numbers of clusters for the first and second sets while using a similarity threshold of 0.4. For the protein sets, the results displayed in Table 5.3 show that SpCLUST, for all the sets, only returned close number of clusters to the expected ones while CD-HIT-Protein and UCLUST produced the exact number for the first, second, and fourth sets with a similarity threshold equal to 0.4, while DACE produced close numbers for the first three sets only.

Given these results, it can be noticed that for the real data sets some tools in some cases give equal or better quality clustering than SpCLUST. However, it is also important to highlight the high impact of the similarity threshold's choice on the quality of the clustering. For example, for the genomic sequences of the second set, UCLUST returned 6 clusters instead of 7 when given a similarity threshold equal to 0.4. On the other hand, it returned 19 and 40 clusters for similarity thresholds equal to 0.6 and 0.9 respectively.

Therefore, for sequence sets with an unknown degree of similarity and an unknown expected number of clusters, SpCLUST remains a better choice because it returns exact or close number of clusters in all cases and without any prior knowledge about the input sequences. Moreover, for the highly similar protein sequences of the third set SpCLUST returned 4 clusters instead of 5, using BLOSUM62's matrix while CD-HIT-Protein and UCLUST put all the sequences in one cluster even with a similarity threshold equal to 0.9.

In summary, for the most divergent sets of the artificially mutated data sets, only SpCLUST gave the exact or close number of clusters, the results of the other tools were very far from the expected ones. In the less divergent simulated and real sets, SpCLUST gave better or at least good results when compared to the other tools. In the next section, the clusters' contents in these experiments are compared and discussed.

5.4.2.2/ Analysis of the clusters' contents

In this section, the contents of the clusters, returned by the selected clustering tools, are compared in order to evaluate their accuracy. To evaluate the degree of similarity between a given clustering and the expected one, the ARI metric was selected. We recall that the ARI computes a similarity measure between two clusterings, the predicted and the true clusterings, by considering all pairs of samples and counting pairs that are assigned in the same or different clusters. It requires the knowledge of the correct cluster which is the case in the previous described experiments. Therefore, the ARI is a good fit in our case since it only requires, for its calculation, the labeling of the clusters' elements according to which cluster they belong in the perfect clustering.

Table 5.4 displays for each simulated data set, the Adjusted Rand Index calculated between the known exact clustering and the one returned by each tool. Only the clusterings that had the correct number of clusters or a very close one are considered because the remaining clusterings have an Adjusted Rand Index close to 0. Based on the values of the computed Adjusted Rand Index, and beside the fact that SpCLUST was the only tool to cluster well all the data sets, the average Adjusted Rand Index for the clustering of the 16 sets was equal to 0.805. Therefore, it can be stated that the proposed module, SpCLUST, performed well, compared to the other tools, and delivered a good overall clustering quality. In contrast, the other clustering tools returned good results in only $1/3$ of the studied cases. Therefore, Even if for $1/3$ of the cases the average of the displayed Adjusted Rand Indexes for certain tools is better than SpCLUST's average index, when considering the remaining $2/3$ cases, where this index falls to nearly zero for the other tools, their overall index averages are way below SpCLUST's average index.

In addition, it can be noticed, from the data in Table 5.4, that CD-HIT-Protein was the

	CD-HIT		DNACLUST	UCLUST		SpCLUST		
Similarity threshold or scoring matrix	0.6	0.4	0.6	0.6	0.4	DNAFULL	PAM250	BLOSUM62
ND3 simul. set S1	-	-	-	-	-	0.082	-	-
ND3 simul. set S2	-	-	-	-	-	0.637	-	-
ND3 simul. set S3	-	-	1	1	0.476	1	-	-
ND3 simul. set S'1	-	-	-	-	-	0.282	-	-
ND3 simul. set S'2	-	-	-	-	-	0.734	-	-
ND3 simul. set S'3	-	-	1	0.778	0.741	0.913	-	-
Prot. simul. set S1	-	-	-	-	-	-	0.406	0.7
Prot. simul. set S2	-	-	-	-	-	-	0.933	0.933
Prot. simul. set S3	1	0.778	-	0.876	0.778	-	0.992	0.992
Prot. simul. set S'1	-	-	-	-	-	-	0.438	0.984
Prot. simul. set S'2	-	-	-	-	-	-	0.933	0.643
Prot. simul. set S'3	1	0.778	-	0.945	0.778	-	0.968	1
Clp. simul. set S1	-	-	-	-	-	0.406	-	-
Clp. simul. set S2	-	-	-	-	-	0.797	-	-
Clp. simul. set S3	-	-	-	1	1	0.783	-	-
Clp. simul. set S'1	-	-	-	-	-	0.368	-	-
Clp. simul. set S'2	-	-	-	-	-	0.517	-	-
Clp. simul. set S'3	-	-	-	1	0.987	0.82	-	-

Table 5.4: Adjusted Rand index for simulated data sets clustering

only tool to return the exact clustering for one of the least divergent sequences' sets of protein, S3. However SpCLUST gave a nearly perfect clustering for this set, with a scored Adjusted Rand Index of 0.992, and using either scoring matrices. As for the other proteins sets, SpCLUST returned the exact clustering for one of them and for the rest of the data sets it gave good quality clustering results having an Adjusted Rand Index varying between 0.7 and 0.992 using either PAM250 or BLOSUM62.

For the genomic sets, on the one hand, DNACLUST and UCLUST succeeded in returning a perfect clustering for the least divergent sets, S3 and S'3. But on the other hand, SpCLUST also performed well and returned clusterings with Adjusted Rand Indexes varying between 0.783 and 1 for these same sets. But although the tools returned very good results, in the case of the least divergent genomic or protein sets they require a user intervention to choose the adequate similarity threshold.

Moreover, since SpCLUST does not rely on any user input identity parameter, it outperforms the other tools in the case of clustering highly divergent data sets. Conversely, the other tools proved to be highly accurate in clustering convergent sets, and at least one of these tools succeeded in finding the true clustering for each one of the least divergent sets.

The mentioned sequence divergence is illustrated in Table 5.5, using Levenshtein distance [lev,] which counts the number of characters insertions or substitutions between two strings. A random starting sequence was chosen from each group of reference sequences and the Levenshtein distance was calculated between this sequence and a ran-

domly chosen mutation of this same sequence from each set, S1 to S'3. The values displayed in Table 5.5 show how the Levenshtein distance decreases from sets S1 and S'1 to sets S3 and S'3. The distances for the other reference sequences in each set should be close to the calculated ones since they were generated using the same mutation criteria.

	S1	S'1	S2	S'2	S3	S'3
ND3	167	178	147	119	41	38
Prot	186	167	158	137	36	44
Clp	120	81	77	56	20	17

Table 5.5: Levenshtein distance between the original and mutated sequences

Finally, although SpCLUST's results do not seem good in the case of the ND3 sets S1 and S'1 (Adjusted Rand Indexes equal to 0.082 and 0.282), a logical reason might explain this phenomenon: in fact, contrary to the other sets which originate from sequences of different genes, the ND3 sets derive from 4 initially divergent sequences of the same ND3 gene. Therefore, the simulated mutations might randomly cause the descendent sequences to converge or make some of the descendent sets highly diverge from the others. In this last case, the distance between the resulting mutated genes might make them look like being, more likely, part of two different clusters instead of four: one cluster containing the descendents that either re-converged towards each other or reached a closely equal distance between each other, and another cluster containing the remainder of the sequences that diverged more from the others. Indeed, the clusters contents, of the clustering of ND3's S1 and S'1, supports the presented theory: for both sets, one cluster contains a certain number of descendants from the same original sequence and the other cluster contains all the remaining sequences.

Table 5.6 presents, for each real data set, the Adjusted Rand Index calculated between the true and the predicted clusterings. SpCLUST was the only tool that was able to return acceptable clusterings for all the real data sets. It produced good overall clustering quality for all the real data sets whereas the other clustering tools failed to return acceptable results for at least one of the sets. Moreover, it can be noticed that SpCLUST outperforms CD-HIT and UCLUST in the first 3 genomic sets and returns better quality clusterings. For the protein sets, CD-HIT and UCLUST gave better quality clusterings than SpCLUST and DACE gave the lowest quality clusterings. Another remarkable phenomenon is that, in contrary to the tests results on the simulated data sets, CD-HIT and UCLUST performed, in general, better than SpCLUST on the real data sets. In fact, a closer look at the data sets shows that 2 out of the 4 sets are very convergent. In fact, CD-HIT and UCLUST both delivered either a perfect or a good clustering for the genomic sets 3 and 4 as well as the protein set 4 with a similarity threshold equal to 0.9. Moreover, for the protein sets 1 and 2, CD-HIT and UCLUST also produced better results than SpCLUST, and for a similarity threshold of 0.6. This shows that the first couple of sets' clusters contents are

	CD-HIT			UCLUST			DACE		SpCLUST		
Similarity threshold or scoring matrix	0.9	0.6	0.4	0.9	0.6	0.4	0.9	0.4	DNA.	PAM.	BLOS.
Set 1 genomic	-	-	-	-	-	0.359	-	-	0.386	-	-
Set 2 genomic	-	-	-	-	-	0.356	-	-	0.47	-	-
Set 3 genomic	0.68	-	-	0.68	-	-	-	-	0.786	-	-
Set 4 genomic	1	-	-	1	1	-	-	-	0.869	-	-
Set 1 protein	-	0.974	0.58	-	0.958	0.712	-	0.409	-	0.625	0.493
Set 2 protein	-	0.928	0.658	-	0.914	0.772	-	0.327	-	0.241	0.354
Set 3 protein	-	-	-	-	-	-	0.205	-	-	0.206	0.456
Set 4 protein	1	1	1	1	1	1	-	-	-	0.869	0.869

Table 5.6: Adjusted Rand index for real data sets clustering

also not very divergent like it was the case in most of the simulated data sets.

Appendix V presents some tables illustrating SpCLUST's clustering contents. Tables 1 and 2 show SpCLUST's clustering contents for the genomic real data sets 2 and 4 that scored Adjusted Rand Indexes equal to 0.47 and 0.869 respectively. Each column in these tables holds an SpCLUST cluster's content. The cells sharing the same color hold sequences that should be in the same cluster. Thus, looking to Table 1 shows that SpCLUST successfully isolated in one cluster, the MC1R sequences that belongs to a gene coming from a different chromosome than all the other sequences is this set. The other clusters contain mainly a mix of 2 or 3 true clusters, and not a random shuffle of elements. This is caused by the fact that elements of these clusters may share by chance a certain percentage of similarity, without presenting the same characteristics or reflecting a real homology. Similarly, looking to Table 2 shows that SpCLUST perfectly isolated Zika virus' sequences from Influenza's. The other clusters also tend to be perfect and only 3 Influenza's H2N2 sequences, that might be a bit too divergent from their other relatives, were clustered separately. It is the same for the set of protein sequences, as shown in Table 3.

Since SpCLUST scored the lowest indexes, 0.386 and 0.354, thus the worst clustering quality, for the first real genomic set and the second real protein set, a further investigation was undertaken for these two sets. The quality of their clustering was evaluated again using another metric: the *purity* of a cluster [Schulte im Walde, 2003]. Table 5.7 shows the purity of the clusters obtained for the first real genomic set and the second real protein set: the clustering of the first set had three pure clusters out of four and the clustering of the second one had two pure clusters out of four and a third cluster with a high purity. Therefore, the resulting low adjusted Rand Indexes do not come from having a bad quality clustering but rather from the merger of two or three clusters with similar elements from the perfect clustering. These results are consistent with the previous analysis of the clustering results of these sets.

Finally, it can be concluded that based on the clusters' contents quality interpreted in

	C1	C2	C3	C4
Set 1 Genomic	1	0.38	1	1
Set 2 Protein	1	1	0.32	0.82

Table 5.7: Clusters purity

this section, SpCLUST presents a better tool for clustering highly divergent data sets, while the other tested tools remain a good choice for more convergent sets, but only in the case of a good similarity threshold choice. However, choosing a good similarity threshold is usually not trivial and requires some knowledge about the data set contents, or the expected number of clusters, which is almost never available in practice for newly discovered sequences. Meanwhile, SpCLUST does not need any user intervention and it offers an innovative tool for clustering new or unknown, genomic or protein, sequence sets.

5.4.2.3/ EFFECT OF SEQUENCES ALIGNMENT QUALITY ON CLUSTERING RESULTS

Since, in the proposed module, the result of the input sequences alignment represents the starting point of the clustering process pipeline, the quality of that alignment, as mentioned in Section 5.3, might impact both the process running time and the clustering accuracy. In this section, the effect of the alignment quality on SpCLUST's results is analyzed on the previously described real data sets.

The Levenshtein distance will be used to compute the distance between the same sequence in two alignments: a fast and a normal one. The ratio between the computed Levenshtein distance and the length of the sequence represents a normalized distance value going from 0 (for exactly similar sequences) to 1 (for completely distinct sequences). Thus, the distance between two alignments can be defined as the average of the distances ratios for all the sequences in the alignment. Table 5.8 shows the distance between the normal alignment and the fast alignment for the genomic and protein mixes of the real data sets. While the second genomic set has the biggest distance, the third genomic and protein sets have identical alignments with the fast and normal alignment.

	Set 1	Set 2	Set 3	Set 4
Genomic	0.1352	0.3	0	0.1473
Protein	0.1424	0.0878	0	0.0368

Table 5.8: Distance between normal and fast alignments

Starting with the cases where the distance between the alignments did not affect the clustering results. In the case of the third mix from the real data sets, containing sequences from Influenza viruses variants, the fast and the normal alignment were identical and they both produced the same clustering. In the case of the fourth mix from the real

data sets, containing mixed sequences from Influenza and Zika viruses, the fast and the normal alignments were slightly different, their normalized distance was equal to 0.1473. However, the SpCLUST's clustering result, using a fast alignment, was similar to the one using a normal alignment, for both genomic and protein sets of this mix. This shows that even with a small normalized distance between two alignments, the quality of the clustering was not affected for these sets. Conversely, for the other sets, and although some distances between their alignments were smaller than 0.1473, it made a difference in the clustering quality.

	Set 1 gen.	Set 1 prot.	Set 2 gen.	Set 2 prot.
SpCLUST normal align.	0.053	0.568	0.41	0.38
Perfect clustering	0.289	0.404	0.383	0.386

Table 5.9: Adjusted Rand Index - Fast alignment

Table 5.9 displays the Adjusted Rand Index between the clustering produced from a fast alignment and those produced from a normal alignment. It also shows the Adjusted Rand Index between the clustering produced from a fast alignment and the true clustering. Fast alignment produced a moderately dissimilar clustering in 3 out of 4 cases when compared to the clustering produced using a normal alignment. Moreover, the calculated Adjusted Rand Index between the clustering, produced from a fast alignment, and the true clustering reflects, in 3 cases out of 4, a slight deterioration in this clustering quality, compared to the one done with a normal clustering. This can be seen by comparing the current values of the Adjusted Rand Index with those in Table 5.6. Appendix V, Table 4 shows, for the second genomic real data set, the SpCLUST clustering produced from a fast alignment and can be compared to the clustering produced from a normal alignment and displayed in Table 1.

Based on the presented results, and knowing that the used viruses' genes sequences (in sets 3 and 4) are much smaller in size than the used mammals genes sequences (in sets 1 and 2), it can be said that a small distance, between a normal alignment and a fast one, does not affect the clustering results in the case of relatively small sequences. Conversely this distance, although being small, slightly impacts the quality of the clustering for larger sequences.

5.5/ CONCLUSION

In this chapter, an efficient and fast clustering package for potentially divergent nucleotide sequences is proposed. This package is based on the Python module presented in [Bruneau et al., 2018] which uses an unsupervised learning method to produce the clustering. However, the new package offers many improvements over the old one such

as enhanced performance due to its implementation in C++ and its parallelization with MPI. A performance comparison between the original package and the new one shown a speed-up ranging from 37.9X to 44.6X when performing a high quality alignment and up to 126X when performing a fast alignment. Moreover, two additional substitution matrices for the distance matrix calculation, PAM250 and BLOSUM62, were added to the package which extends its capabilities to cluster protein sequences. The proposed package compiles and runs on both Linux and Windows and can be easily integrated to a GALAXY server.

A comparative study between SpCLUST and some existing and widely used clustering tools, such as UCLUST [Edgar, 2010], CD-HIT [Li et al., 2006], DNA-CLUST [Ghodsi et al., 2011] and DACE [Jiang et al., 2016], was conducted over different sets of simulated and real, genomic and protein, sequences. In contrast with these state of the art tools, SpCLUST does not mainly aim for higher clustering speeds, of highly similar sequences, than its competitors. SpCLUST aims for fast clustering of data sets containing potentially divergent elements, and without any a priori knowledge of the similarity threshold or the number of clusters. The experiments shown that in the most cases SpCLUST gave better or fairly good results, compared to the other tools, in terms of number of clusters and their contents. Moreover, for highly divergent sequences that the other tools were not able to cluster, SpCLUST gave good clustering quality compared to the expected clustering. Finally, unlike the other tools that need a highly influencing similarity threshold parameter input, SpCLUST does not require any user intervention. Further improvements to this GMM-based package are presented in the next chapter.

6

OPTIMIZED SPECTRAL CLUSTERING METHODS

6.1/ INTRODUCTION

Clustering of biological sequences is currently playing a paramount role in linking the huge number of newly discovered sequences to their variants and ancestors. However, current methods can only partially tackle this problem due to its scale and complexity. Recent research has concluded that spectral clustering may represent an efficient tool for biological sequence clustering [Pentney et al., 2005, Paccanaro et al., 2006, Hu et al., 2004] and, to our knowledge, the only tool has been publicly released is our SpCLUST [Matar et al., 2019] that has been introduced in Chapter 5. In Chapter 5, the relevance of using GMM's (Gaussian Mixture Models) for unsupervised clustering of biological sequences was demonstrated through various numerical validation experiments. Contrarily to most of the widely used clustering tools, GMM-based approaches require no user intervention and is well adapted to clustering divergent sequences as well.

One of the main difficulties in studying newly discovered biological sequences resides in that, due to their unknown degree of divergence, neither an accurate selection of the similarity threshold nor the selection of the clusters' centroids are trivial. In such cases, traditional tools, requiring a user-defined similarity threshold, cannot be considered reliable. On the other hand, GMM-based alternatives which do not require any a priori knowledge of an arbitrary similarity threshold, seem to be well adapted to efficiently tackle such problems. GMM's and other finite mixture models [McLachlan et al., 2004] are usually calibrated using an Expectation Maximum (EM) algorithm [Dempster et al., 1977, Wu, 1983, McLachlan et al., 2007] or one of its accelerations [Chrétien et al., 1998, Chrétien et al., 2000, Celeux et al., 2001]. As a reason for its success, GMM offers improved classification performances in several applications where clusters overlap, which has proved key in such fields as biological sequence clustering [Matar et al., 2019], age and gender recognition [Bocklet et al., 2008], real-time segmentation of HD

video [Genovese et al., 2013], etc. However, the use of EM-type algorithms requires expertise due to the well known drawbacks [Biernacki et al., 2016, Biernacki et al., 2003] and computational issues for large and high dimensional data [Shi et al., 2006], and one should always give its preference to reliable packages which carefully address these subtle technical issues.

We recall that the tool presented in Chapter 5 implements the following operations for clustering a set of biological sequences: i- sequences' alignment, ii- pairwise affinity computation of the sequences, iii- Laplacian Eigenmap embedding of the data, and iv- GMM based clustering. The quality of the generated clustering and the performance of this approach is often greatly impacted by the implementation choices made at each stage: the alignment quality, the type of affinity between sequences, the GMM implementation which succeeds or fails to address some of the drawbacks described in [Biernacki et al., 2016], etc. The present chapter investigates how the use of different techniques and their implementation at each stage of the pipeline contributes in accelerating the clustering or improving its quality. The main studied factors are:

- the GMM implementation;
- the sequence alignment tool;
- the calculation of the affinity matrix between the sequences.

Our contribution in this chapter is a new clustering package, called SpCLUST-V2, which improves on its predecessor in many directions [1]. This package is able to accurately cluster biological sequences with various degrees of divergence, and its use does not require any a priori knowledge of an arbitrary similarity threshold. Our package incorporates several new model selection criteria and its better performance is illustrated via several numerical experiments.

The remainder of this chapter is organized as follows. In Section 6.2, our contributions are presented. The experimental protocol is detailed and the numerical evaluations are presented in Section 6.3. Finally, Section 6.4 concludes this chapter by recapitulating our new findings.

[1] and whose latest release can be found at the address https://github.com/johnymatar/SpCLUST-V2 or http://galaxy.ul.edu.lb

6.2/ APPROACH AND METHOD

6.2.1/ THREE WAYS OF IMPROVEMENT

The biological sequences clustering method, presented in [Matar et al., 2019], led to the release of a publicly available package named SpCLUST. This package does not require any identity threshold or centroids as user input. However, further enhancements to its first release remained possible. These enhancements mainly fit into three categories:

1. Performance-wise: an intensive part of the proposed method, i.e. the computation of the GMM, remained in Python. But if this programming language is well suited for fast prototyping, its use is less relevant when targeting the final version of the program, which should be fast and efficient [Müller et al., 2011]. Other GMM implementations, based on lower level programming languages such as C++, might perform faster and significantly improve the overall performance of the method. The alignment stage is another time-costly part, to improve in the case where long sequences or a large amount of data are submitted.
2. Features-wise: SpCLUST allows the user to customize the sequences alignment quality. For instance, it is possible to set the maximum number of iterations in MUSCLE, which affects both the alignment quality and speed, and to choose the desired substitution matrix for pairwise distance calculations. Additional options and features can be added to the package, like some user-customizable statistical parameters for model selection in clustering, or to offer the possibility to input aligned sequences.
3. Algorithmic-wise: the Laplacian Eigenmap, in SpCLUST, is computed from a Random Walk Normalized Laplacian matrix. Studying the effect of using different types of matrices, and offering to the user the ability of choosing between these matrices, is interesting: the Non-normalized Laplacian, the Modularity, and the Bethe Hessian (also called Deformed Laplacian), for instance, are other types of relevant matrices that exist in the literature. Each choice may potentially lead to a different clustering.

The proposed enhancement methods, options, and algorithms are discussed in the next subsections. And since the main objective of SpCLUST remains enhancing the clustering quality, the impact of the implemented options on this latter is discussed in Section 6.3.

6.2.2/ IMPROVEMENTS IN THE GMM PART

The $GaussianMixture()$ function [Gau, b] is the successor of the obsolete $GMM()$ function [GMM,] provided in Python's scikit-learn package [Pedregosa et al., 2011]. It gave

similar results as $GMM()$ when integrated to SpCLUST. This function takes an $m \times n$ matrix as input, where m is the number of features and n is the number of samples. It includes methods that calculate indices, such as the BIC and AIC, that reflect the quality of the GMM. It also allows the user to manually input several parameters including the co-variance type among full (default), tied, diagonal and spherical, together with a seed that affects the initial random distribution. But a detailed profiling of SpCLUST shows that the GMM clustering stage, using the $GaussianMixture()$ function, is the most time consuming stage in the pipeline [Matar et al., 2019].

Another Python scikit-learn function, the $spectral\_embedding()$, implements the spectral decomposition in a different manner [Spe,]. It merges the dimensionality reduction and the sequence clustering phases and takes an $n \times n$ pairwise similarity matrix as input, where n is the number of samples. The (normalized or not) Laplacian matrix computation is embedded in the $spectral\_embedding()$ function. But contrary to the $GaussianMixture()$ function, the $spectral\_embedding()$ one does not provide any method to compute statistical indices of quality.

By comparison, few C++ implementations of the GMM are freely available, but they do not offer as many features and options as the $GaussianMixture()$ function introduced earlier. For instance, a multi-threaded and open source implementation, featuring the diagonal and full co-variance types, is included in the Armadillo C++ library [Sanderson et al., 2016, Sanderson et al., 2017]. But this implementation does not offer any statistical indices for evaluating the quality of the GMM [arm,]. It also requires having the Armadillo library installed as a prerequisite. Another GMM implementation, namely the paperrune one [pep,], only uses the standard C++ libraries, and it also includes the Likelihood model evaluation index.

As can be seen, BIC and AIC indices are missing in the previously mentioned C++ implementations. Thus, to get the same capabilities than the $GaussianMixture()$ or $GMM()$ Python functions while improving performance, the latter have been translated to C++. And to save more computation time, a freely available C++ implementation of the K-Means algorithm [kme,] has been used as part of the GMM pipeline. As the C++ pseudo-random number generator, used in K-Means, is not cross-platform consistent (the rand() function is not the same depending on the platform, and using the same seed will generate different numbers on different operating systems), and in order to preserve the consistency of the results, a custom pseudorandom numbers generation function has been implemented. It is based on Microsoft's rand formula: $(a * seed + c)\%m$ with $a = 214013$, $c = 2531011$, and $m = 2^{31}$. And if no seed is provided by the user, the $seed$ variable is equal to its default value, that is, 0.

The improvements on the GMM part having been presented, we will now discuss the other improvements of the SpCLUST package, by starting with the ones related to the

dimensionality reduction.

6.2.3/ Improvements related to the Affinity matrix and the Eigenmap calculation

In the clustering pipeline, the affinity matrix was until now computed as a Random Walk Normalized Laplacian, which has been proven in [Bruneau et al., 2018] and [Matar et al., 2019] to be relevant for the clustering of biological sequences. However, other interesting matrices have been proposed for spectral clustering [Von Luxburg, 2007, Langone et al., 2011, Saade et al., 2014, Dall'Amico et al., 2019a], such as the Non-normalized Laplacian, Modularity [Langone et al., 2011] and the Bethe Hessian (Deformed Laplacian) [Dall'Amico et al., 2019b]. These matrices are defined as follows:

- **Non-normalized Laplacian:**

  $$L = D - A,$$

  where A is the adjacency matrix between the sequences and D is its diagonal matrix of degrees.

- **Random Walk Normalized Laplacian:**

  $$L^{rw} = D^{-1}L,$$

  where D is the degrees matrix of the adjacency matrix and L is the Non-normalized Laplacian matrix.

- **Modularity:**

  $$M = \frac{1}{K}\left(A - \frac{1}{K}kk^T\right),$$

  where A is the adjacency matrix, k is the degrees vector of A, and K is the total degree of A.

- **Bethe Hessian:**

  $$H_r = (r^2 - 1)I + D - rA$$

  where I is the identity matrix, D is the degrees matrix of the adjacency matrix A, and the constant r is the square root of the average degree of the graph, as suggested in [Saade et al., 2014].

Let us remark the following points concerning these definitions. The Laplacian is a symmetric and positive semidefinite matrix. The Non-normalized Laplacian and the Normalized Laplacian serve respectively in the approximation of the minimization of the RatioCut

and the NCut. The Modularity is a quality function whose high values reveal the possible existence of strong communities. Finally, the Bethe Hessian, also called deformed Laplacian, features a regulator constant r in addition to the previously introduced Laplacian matrices. The performances of all these matrices for clustering will be compared in Section 6.3.

6.2.4/ FURTHER ADDITIONAL FEATURES

In Python's $spectral\_embedding()$ function, the user can choose either the normalized or the non-normalized Laplacian matrix. The former is used by default. Moreover, the dimension of the projection subspace, reflecting the number of resulting clusters can be specified; by default, this parameter is set to 8. Since the $spectral\_embedding()$ function does not offer further exploitable parameters, we did not propose further features inherited from this function.

The paperrune's GMM implementation includes a method to compute the likelihood of the model. Therefore, it is possible to automatically choose the best clustering by maximizing this likelihood. This is achieved by performing several iterations as illustrated in Figure 6.1. The given number of clusters is modified at each iteration, and it ranges between 1 and the number of sequences.

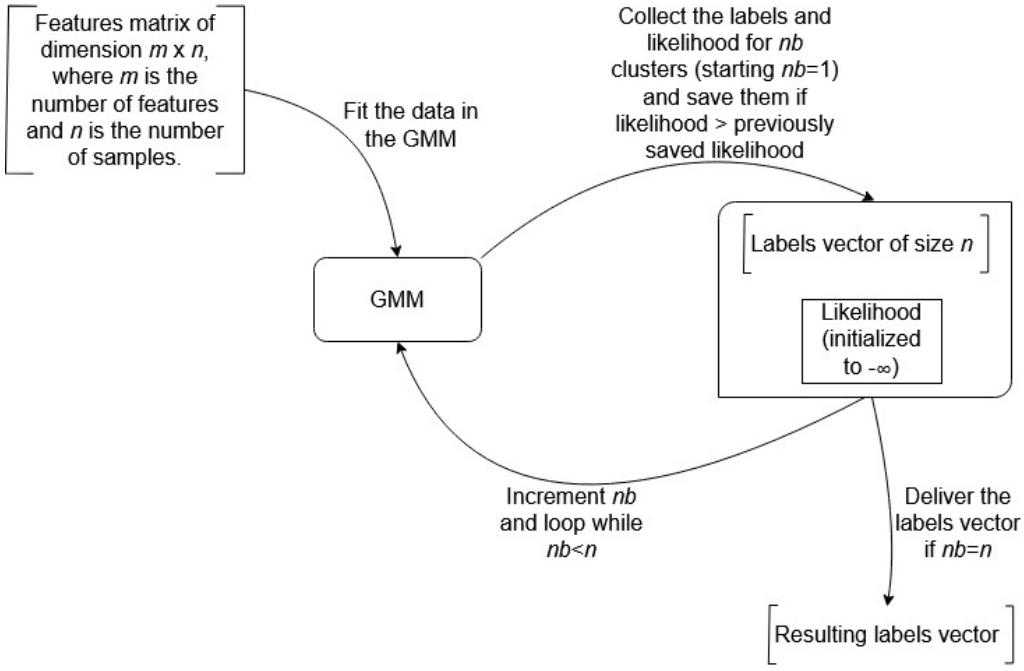

Figure 6.1: Choosing the best clustering based on maximum likelihood.

By comparison, our porting of Python's GMM function to C++ can compute the BIC and AIC criteria which assess the quality of the model. Three different algorithms are proposed to exploit both criteria:

1. The first one is similar to the previously presented algorithm in Figure 6.1. It is applicable by simply substituting the maximum likelihood by the lowest BIC (or AIC).

2. The second approach consists of iterating the first one a user-defined number of times, with a different random seed at each iteration. Let us recall that the random seed impacts the initial random distribution of the centroids, leading to a potentially different clustering for each seed. Following these iterations, the clustering that scores the maximum number of occurrences is selected. The counting procedure of the occurrences of each clustering distinguishes between same clustering with different labelling and different clustering. Figure 6.2 illustrates this method. Its computation time, compared to the previous one, is proportional to the chosen number of iterations. Moreover, this algorithm requires a larger amount of memory, since it saves the labels vector for the resulting clustering at each iteration. Therefore, it requires a certain memory size if the input dataset and the chosen number of iterations are both large.

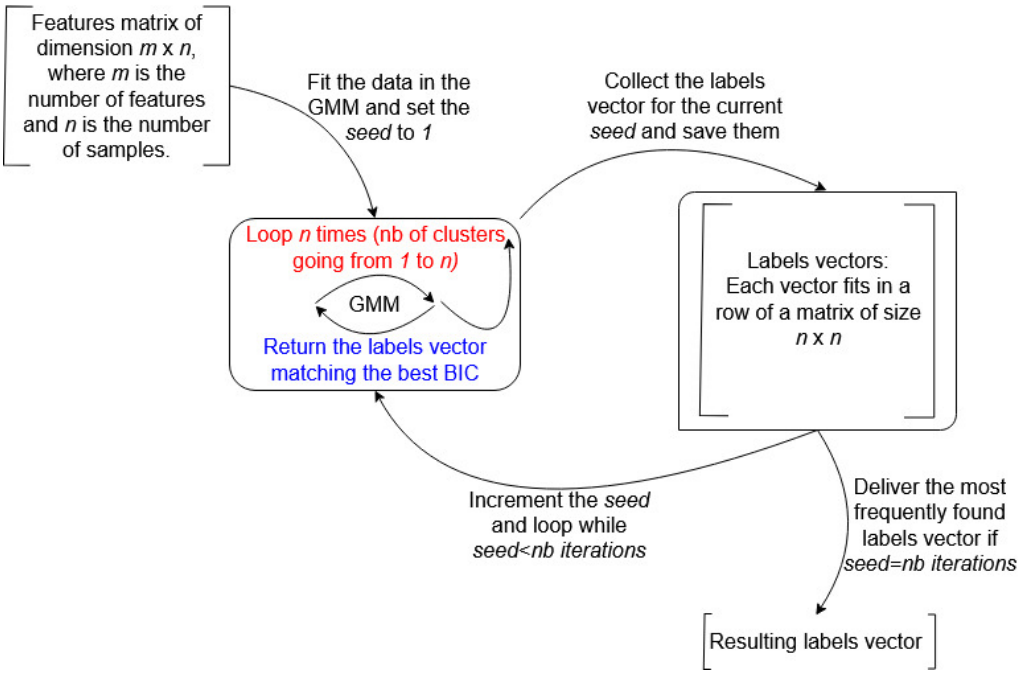

Figure 6.2: Choosing the best clustering based on the occurrence frequency.

3. The third algorithm is similar to the second one. It successively clusters the sequences using different seeds, but just keeps in memory the designated best clustering (e.g., the one that scores the best BIC). Moreover, in order to reduce the execution time of this algorithm, an additional parameter can be defined to stop the iterative process before reaching the chosen number of iterations, if no BIC improvement is detected after a certain number *noImp* of consecutive iterations. Figure 6.3 illustrates this algorithm that requires less computation in the case where the stop

condition is fulfilled prior to reaching the chosen number of iterations.

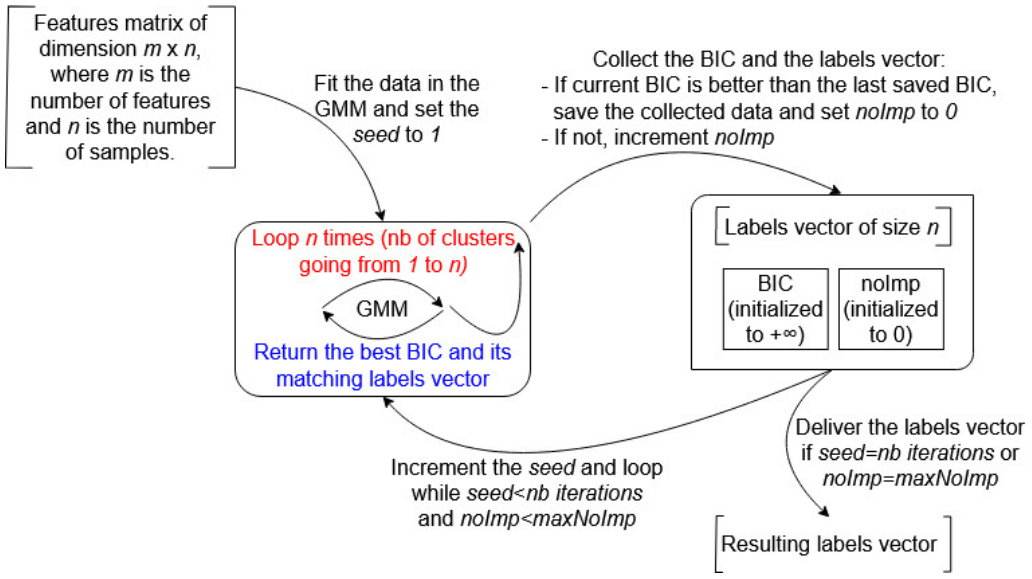

Figure 6.3: Choosing the best clustering based on the best reached BIC.

The user can choose any of the three proposed algorithms to cluster a given set of biological sequences.

6.3/ EVALUATION OF THE NEW PACKAGE

The proposal is evaluated in this section, according to its three main parameters: the GMM variation, the alignment tool, and the affinity matrix. The experimental protocol is first described, then the clustering results are detailed and discussed, by using external and internal validation methods or indices.

6.3.1/ EXPERIMENTAL PROTOCOL

6.3.1.1/ THE DATASETS

Three real biological sequence dataset have been considered to evaluate the proposal:

- A first set of 78 complete genome sequences, belonging to $HIV-1$ type B virus samples identified in Cyprus, and downloaded from the Los Alamos National Laboratory's website[2].

[2]https://www.hiv.lanl.gov/components/sequence/HIV/search/search.html

- A second set of 100 genomic sequences, belonging to the $NADH$ dehydrogenase 3 (ND3) mitochondrial gene, from a collection of Platyhelminthes and Nematoda species.

- A third set of 24 different nucleoprotein (NP) sequences, belonging to the strain A/H1N1 of the $Influenza$ virus, and downloaded from NCBI's Influenza Virus Database[3].

Table 6.1 shows a brief description of the first three datasets which contain each a single type of sequences. The statistics on the sequences were retrieved from the output of MUSCLE [Edgar, 2004]. The pairwise similarity, between the sequences of each dataset, was computed using MatGAT [Campanella et al., 2003] which calculates the similarity after using the Myers and Miller global alignment algorithm [Myers et al., 1988].

Dataset	Seqs count	Max length	Avg length	Min similarity %	Max similarity %	Avg similarity %
HIV	78	8272	8167	86	99.4	89.6
NADH	100	369	341	46.2	99.7	62.8
Influenza	24	498	498	97.4	99.8	98.8

Table 6.1: Statistical description of the datasets.

Moreover, additional datasets were considered in order to evaluate the capability of the tool to: i- correctly group different pathogens that could be collected from a similar region of infection or from different regions, ii- accurately identify and group viral genomes that could have received a gene segment by horizontal gene transfer (HGT), following the occurrence of a common host cell incubation. For these purposes, a SARS-COV complete genome along with the complete set of segments belonging to the genomes of Influenza strains A and D were retrieved from viruSITE[4]. Then, five additional SARS-COV genomes were generated by simulating the mutation of 2% of the original genome and by artificially inducing a similar ratio of random gaps and insertions. Similarly, nine additional genomes were obtained from each one of the complete genomes of Influenza A and Influenza D that were assembled using the retrieved segments.

In order to simulate a horizontal gene transfer: i- two random gene segments were extracted from an HIV genome (from the first dataset), ii- two Influenza A, two Influenza D, and two SARS-COV genomes (among the previously generated ones) were selected: the root sequence and one of its direct descendants, iii- both extracted HIV gene segments were inserted in between segments 1 and 2, and segments 6 and 7 in the selected Influenza A and Influenza D genomes, iv- both HIV genes segments were also inserted at two random locations in the selected SARS-COV sequences. The six newly generated

[3] https://www.ncbi.nlm.nih.gov/genomes/FLU/Database/nph-select.cgi
[4] http://www.virusite.org/archive/2021.1/genomes.fasta.zip

genomes replaced the original ones in the Influenza and SARS-COV datasets. Although a similar gene transfer is unlikely to happen in-vivo, yet it remains theoretically possible in-vitro, for example in [Khurana et al., 2007], it was shown that a human HeLa cell can simultaneously incubate and produce both Influenza and HIV-1 viruses.

The 26 resulting genomes from the previous simulations (10 Influenza A, 10 Influenza D and 6 SARS-COV), in addition to 9 HIV genomes randomly selected from the first dataset, have been used to assemble four additional biological datasets, containing each a mix of different genomes:

- A fourth set of 20 complete genomes, consisting of the 10 Influenza A and the 10 Influenza D genomes.
- A fifth set of 26 genomes, including the 20 Influenza genomes from the fourth set and the 6 SARS-COV genomes. This set and the fourth one contain pathogens that infect the same area.
- A sixth set of 29 genomes, including the 20 Influenza genomes from the fourth set and the 9 HIV genomes that were randomly picked from the first set.
- A last set of 35 genomes, including all the genomes of the two previous sets. This set and the sixth one contain pathogens that have different zones of infection.

All these datasets are also available on the SpCLUST-V2 GitHub repository[5].

6.3.1.2/ THE REFERENCE CLUSTERING

Since a clustering golden truth is not available for the first three sets of sequences, a phylogenetic tree, showing the evolutionary relationship among the sequences of each set, is used for assessing the validity of the clustering results. Indeed, there are many tools that, given an aligned set of sequences, can build the phylogenetic tree of these sequences. In this work, the tree for each set of data was built according to the following procedure:

1. MUSCLE [Edgar, 2004] computed the sequences' alignment.
2. PhyML 3.0 [Guindon et al., 2010] generated the phylogenetic tree. The automatic model selection, based on the likelihood criteria, was selected. This selection, provided by SMS [Lefort et al., 2017], was set to use the BIC (Bayesian Information Criterion).

[5] https://github.com/johnymatar/SpCLUST-V2

3. The resulting phylogenetic tree was visualized using PRESTO (Phylogenetic tReE viSualisaTiOn)[6].

Based on the previously generated phylogenetic trees, each clustering, produced in the next experiments, is assessed individually. Since it is possible in each clustering to identify valid subclusters, it is not fair to assess all the clustering methods by using a single unified reference per set. Therefore, we have defined a custom algorithm for assigning a reference for each produced clustering. The algorithm takes as input the considered clustering and the phylogenetic tree and produces the reference clustering. It consists of the following steps:

1. From the given clustering, the elements of the phylogenetic tree are assigned labels as illustrated in Figure 6.4. The labels indicate to which cluster each sequence belongs in the given clustering. For example, in Figure 6.4, the clustering produced four clusters: cluster 1 to 4 are represented by the labels *, #, - and + respectively.

2. The depth of the phylogenetic tree (TD) is computed, a $counter$ is initialized to $TD-1$ and at each iteration it is decremented by 1 till 0.

3. On each iteration, for each inner node that has a depth equal to the $counter$, the following cases are possible:

    1. if all the first level descendants of the node are leaves, a cluster consisting of these leaves is formed. The newly formed cluster is labelled according to the dominant label, the label that occurs the most among the cluster elements. If no dominant label was found, i.e. two labels have the same high number of occurrences, the undefined label is attributed to the cluster.

    2. if the first level descendants of the node include a leaf and at least one already formed cluster, the leaf is added to the cluster that is the closest to it. The cluster is relabelled according to the dominant label between its elements.

    3. if the first level descendants of the node include at least two clusters and: a- two adjacent clusters have the same label, they are merged. b- one of the clusters is labelled as “undefined”, it is merged with an adjacent cluster and the resulting cluster is relabelled according to the dominant label between its elements. c- two clusters have different labels, they are not modified. d- one of these clusters is small (less than 4 elements) and is surrounded by two larger clusters having the same label, the small cluster is merged with its surrounding clusters because it is considered to be just noise in the cluster.

[6] http://www.atgc-montpellier.fr/presto/

4. After the final iteration, if there is still clusters with undefined labels, they are assigned new labels. If two or more clusters have the same label, they are also assigned new labels.

Figures 6.4 and 6.5 illustrate how a reference clustering is generated according to the algorithm described above. In the first sub-figure of Figure 6.4 the elements of the phylogenetic tree are assigned labels (*, #, - or +) which indicate to which cluster each sequence belongs in the given clustering. The depth of each node in the tree is also displayed. In this example, the depth of the tree (TD) is equal to 6. After this initialization step, the iterative process starts with the inner nodes at $depth = TD - 1$. The second sub-figure of 6.4 illustrates the first iteration of the algorithm. In this example, there is only one inner node with a $depth = 5$. It contains two leaves/sequences (Elt 11 and Elt 12). Both sequences belong to the third cluster. Therefore, a cluster containing both sequences is formed and labeled as "Cluster 3" in the reference clustering. This new cluster is represented by a red rectangle in Figure 6.4. Figure 6.5 illustrates the remaining iterations. At the second iteration with inner nodes of $depth = 4$, three new clusters are created. The first one consists of Elt 1 and Elt 2 and is labeled as "Cluster 1" because both of its sequences belong to the first cluster. The second cluster is created in the same way as the previous one. The third new cluster consists of Elt 14 and Elt 15 which belong to different clusters and thus there is no dominant label in this cluster. For this reason this cluster is labeled as "Undefined". It can also be noticed that Elt 3 was added to "Cluster 3" and since "Cluster 3" is still the dominant label in this cluster, its label was not changed. Figure 6.5 displays the next three iterations and then the iterative process stops at the root node ($depth = 0$). In this example, the resulting reference clustering consists of three clusters: the first two are homogeneous but the third one contains sequences belonging to three different clusters in the given clustering. However, six of its nine sequences belong to the same cluster and thus their dominant label is assigned to this cluster.

6.3.1.3/ THE EXPERIMENTS

The first set of experiments aims to compare the GMM implementations presented in Section 6.2. The three first data were used for this set of experiments. In this evaluation, after the alignment stage using MUSCLE, the similarity matrices and the Eigenmaps are calculated using the same algorithms used in SpCLUST. The clustering is then computed using one of the following methods:

- Python's $GaussianMixture()$ function that is embedded in the former version of SpCLUST.
- The algorithm introduced in Figure 6.1, which uses paperrunes's C++ GMM imple-

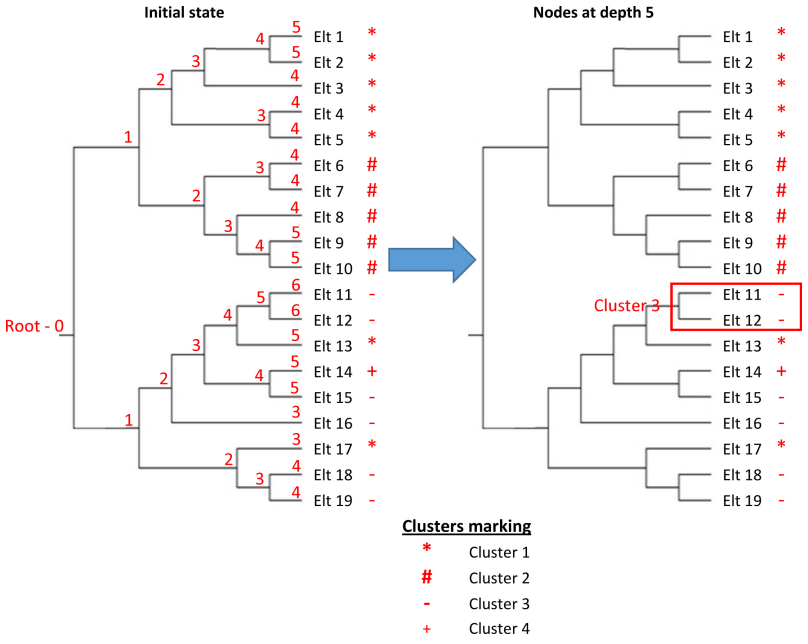

Figure 6.4: Initial state and first iteration.

mentation.

- Python's $spectral\_embedding()$ function using the Normalized Laplacian matrix, as in SpCLUST.
- The three algorithms presented in Section 6.2 that use the C++ GMM implementation we translated from Python's $GMM()$ function. The method described in Figure 6.1, in which the maximum likelihood is replaced by the best BIC, will be named "Fast". The one from Figure 6.2 will be referred as "MostFreq", and will be tested during 500 iterations. Finally, the algorithm from Figure 6.3 will be named "BestBIC". It will be tested during 100 iterations, with a "no improvement" stop parameter set to 70 consecutive iterations.

The used datasets will also be clustered using UCLUST and CD-HIT, which are the best competitors to SpCLUST [Matar et al., 2019].

The goal of the second set of experiments is to measure the impact of the alignment tool, by replacing MUSCLE with other popular software. This assessment covers both the clustering quality and the efficiency of the resulting pipeline. These experiments were conducted as follows:

1. The three first sets of sequences are aligned using the following packages: MUSCLE, MAFFT [Katoh et al., 2013], DECIPHER [Wright, 2015], and CLUSTALX [Larking et al., 2007].
2. The resulting aligned sets are clustered using the GMM implementation of

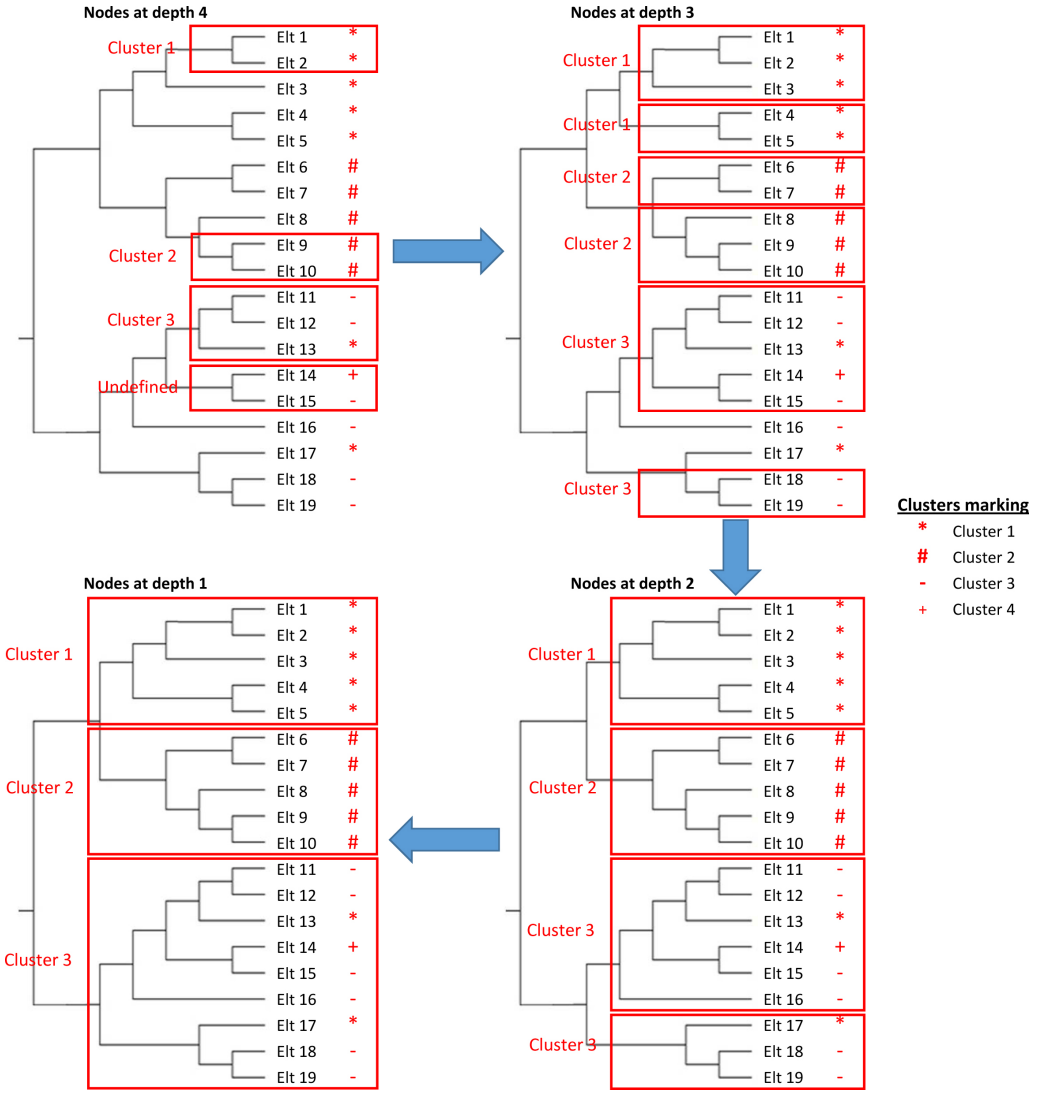

Figure 6.5: Clusters identification and final state.

SpCLUST-V2.

The third set of experiments in this protocol consists in evaluating the clustering quality with different types of affinity matrices. The Non-normalized Laplacian, Modularity, and Bethe Hessian will be compared to the Normalized Laplacian already used in SpCLUST. This set of experiments was conducted on the three previously used datasets.

The release version of SpCLUST-V2 will be based on the results in the aforementioned experiments. Moreover, additional internal clustering validation methods that do not rely on a ground truth clustering, will be also considered. The main advantage of the internal validation methods is that they avoid any potential artifacts related to the generation of the reference clustering.

Finally, a last set of experiments that evaluate the capability of SpCLUST-V2 to cluster a set consisting of different pathogens, was conducted over the last four datasets.

6.3.2/ THE EXPERIMENTAL RESULTS

6.3.2.1/ THE GMM IMPLEMENTATION IMPACT ON THE CLUSTERING

In the first set of experiments, the presented GMM implementations and the proposed algorithms detailed in Section 6.2 are evaluated. We recall that the computation of the Laplacian Eigenmap is embedded in the $spectral\_embedding()$ function. Conversely, for SpCLUST-V1, the Eigenmap is computed using functions from the numpy linear algebra library. For the remaining C++ implementations of the GMM, an implementation[7] of Jacobi's Eigen solving algorithm is used. The datasets are also clustered using state-of-the-art UCLUST and CD-HIT. In order to cover a wide range of similarities, the identity thresholds chosen for UCLUST and CD-HIT ranged between 0.5 and 0.99, with a step of 0.1 in the $[0.5 - 0.8[$ interval and a step of 0.01 in the $[0.8 - 0.99[$ one. For any identity threshold lower than 0.8, CD-HIT failed to cluster the data. For the sake of comparison, only produced results having a number of clusters close to the ones from SpCLUST were considered.

Figures 6.6, 6.7, 6.8, 6.9, and 6.10 show the labeled elements on the phylogenetic trees. To improve the legibility of the large HIV and $NADH$ phylogenetic trees, each one of them was split into 2 sub-trees. Next to each sequence in the tree, there are labels. Each one of these labels indicate to which cluster this sequence belonged in the clustering obtained with a given method. For example, in 6.6, the first element (JF683743) belonged to the fifth cluster in the clustering obtained with SpCLUST, to the second cluster in the clustering obtained with paperrune, etc. The colors of the labels indicate with which tool they were obtained.

To evaluate the quality of each clustering, the degree of similarity between it and the reference must be computed using a relevant metric. Many clustering quality metrics are available in the literature [Wagner et al., 2007, Guyeux et al., 2019]. In this work, the Adjusted Rand Index (ARI) was selected to compute the degree of similarity, because it only requires the labels, and it is able to compare clusterings with different number of clusters. This index ranges between 0 for two completely different clusterings and 1 for two identical ones.

Table 6.2 displays, for each dataset and each clustering returned from the considered methods, the number of clusters in both generated and reference clusterings, and the ARI between them. Note that ARI is omitted in the following three special cases:

1. when a clustering consists of only one cluster;

2. when the number of clusters, formed of singletons, is greater than the half of the

[7] https://github.com/edwardlfh/testv2/tree/master/jacobi

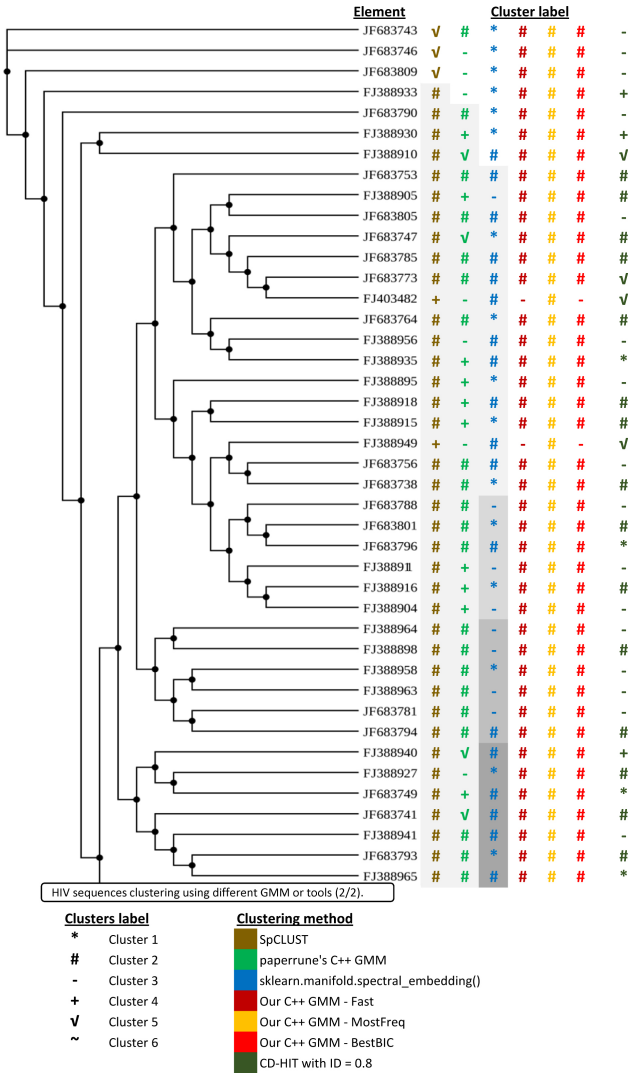

Figure 6.6: HIV sequences clustering using different GMM or tools (1/2).

number of sequences (most of the sequences are clustered as one sequence per cluster);

3. when the labels of adjacent leaves on the phylogenetic tree are very heterogeneous and the resulting clustering does not reflect any correct grouping on the tree.

The clusterings, matching the first special case, will be discussed later according to the properties of the involved dataset. Conversely, those matching the second case are not significant, because the sequences belonging to a same dataset are a priori known to be related.

The MostFreq algorithm scored the best ARI in the case of clustering the HIV set of sequences. The BestBIC version obtained the second rank, followed by the Fast algorithm. The matching between the number of clusters in the reference clustering and the generated clustering shows a better accuracy for the used algorithms. By comparison,

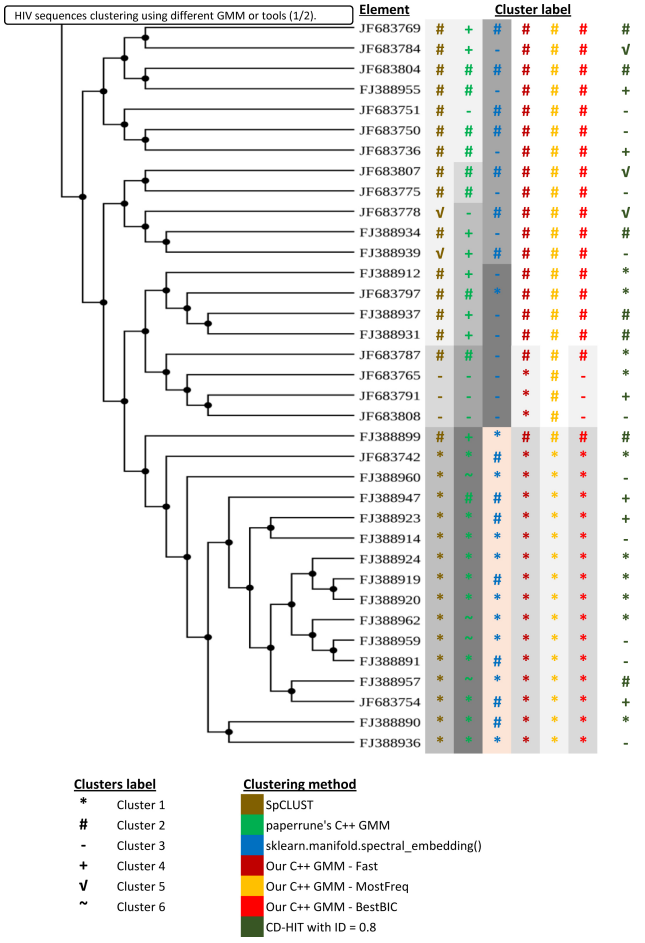

Figure 6.7: *HIV* sequences clustering using different GMM or tools (2/2).

	HIV			NADH			Influenza		
	Nb. Clusters		ARI	Nb. Clusters		ARI	Nb. Clusters		ARI
	ref.	gen.		ref.	gen.		ref.	gen.	
SpCLUST	4	5	0.777	5	4	0.957	4	5	0.932
Paperrune's GMM	6	6	0.236	11	8	0.838	3	3	0.847
sklearn.manifold.spectral_embedding()	7	3	0.119	7	4	0.694	4	3	0.653
Our GMM - Fast	3	3	0.801	4	2	0.804	-	1	-
Our GMM - MostFreq	2	2	0.941	4	2	0.841	-	1	-
Our GMM - BestBIC	3	3	0.828	4	3	0.839	2	2	1
UCLUST (id 0.5)	-	78	-	5	6	0.374	-	1	-
UCLUST (id 0.88)	-	78	-	-	83	-	-	1	-
UCLUST (id 0.89-0.94)	-	78	-	-	86-95	-	2	2	1
UCLUST (id 0.95-0.96)	-	78	-	-	97	-	3	3	1
CD-HIT (id 0.91)	-	66	-	-	90	-	-	1	-
CD-HIT (id 0.92)	-	69	-	-	92	-	1	2	-
CD-HIT (id 0.93-94)	-	71-72	-	-	94-95	-	2	2	1
CD-HIT (id 0.95-0.97)	-	73-75	-	-	97-98	-	3	3	1

Table 6.2: External clustering validation using the Adjusted Rand Index.

SpCLUST produced one additional cluster than the reference clustering, which penalized its score. Conversely, Python's *spectral_embedding*() implementation produced three clusters less than the reference because in the reference clustering non-adjacent clusters on

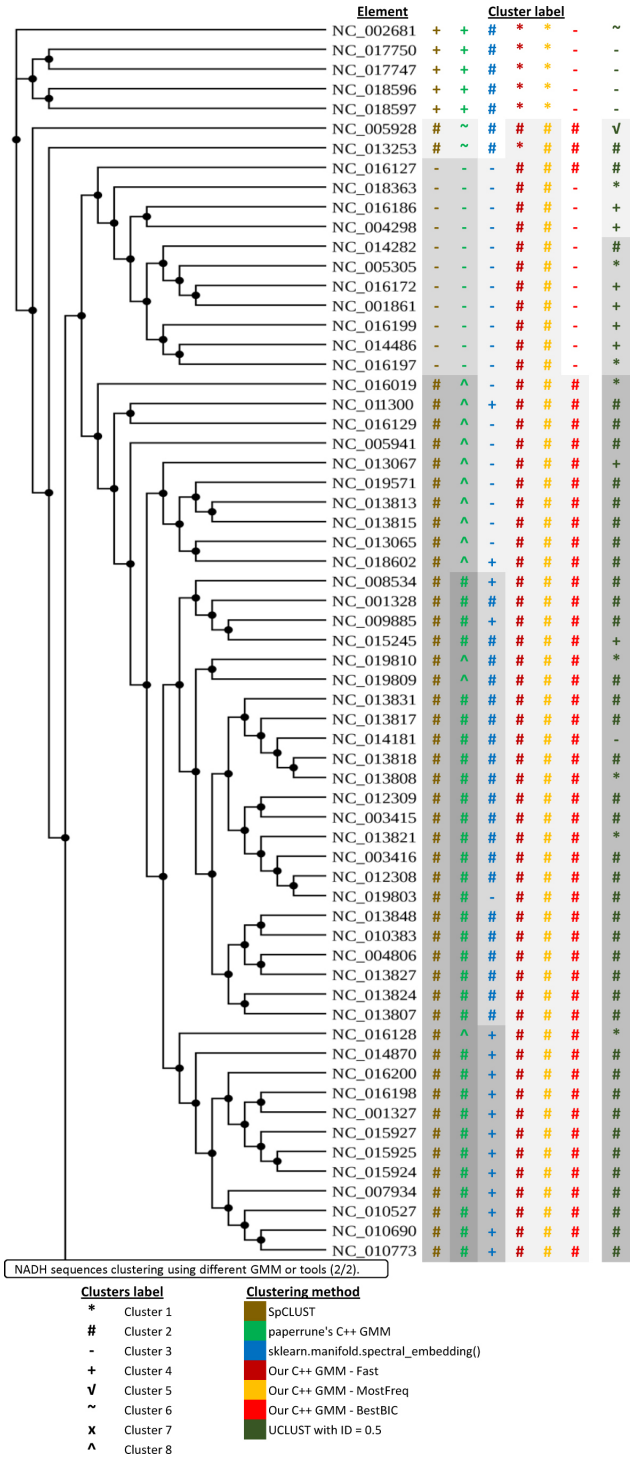

Figure 6.8: *NADH* sequences clustering using different GMM or tools (1/2).

the tree are not merged. Finally, UCLUST and CD-HIT both failed to cluster this set, although its sequences show a minimum similarity of 86% (cf. Table 6.1). Indeed, CD-HIT produced 5 clusters when the similarity parameter was set to 0.8, but these clusters do not reflect any meaningful grouping on the phylogenetic tree, as illustrated in Figures 6.6 and 6.7.

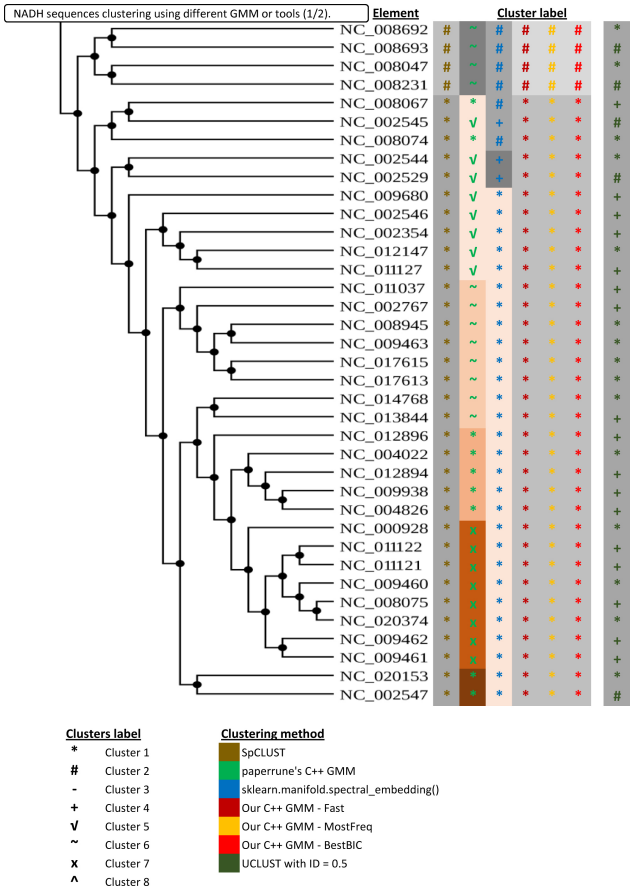

Figure 6.9: *NADH* sequences clustering using different GMM or tools (2/2).

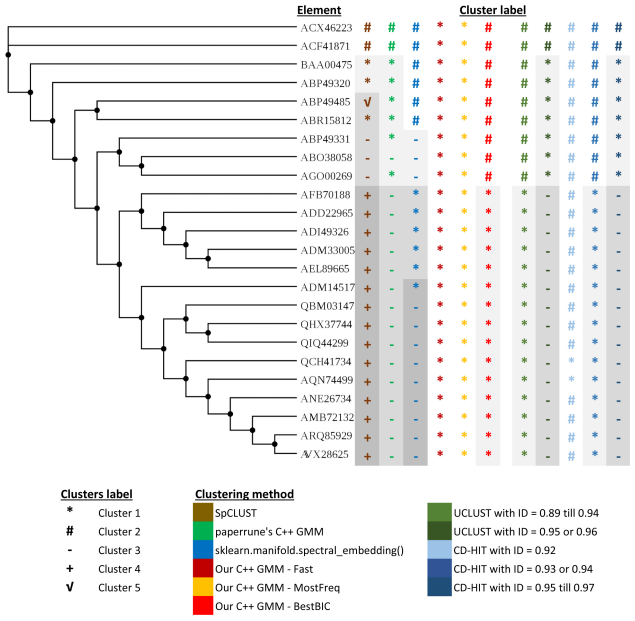

Figure 6.10: *Influenza* sequences clustering using different GMM or tools.

SpCLUST GMM scored the highest ARI values for the *NADH* dataset, followed by our GMM implementation with the MostFreq and BestBIC approaches respectively. Most-

Freq and Fast approaches produced two highly similar clusterings: only one element was labelled differently, as illustrated in Figures 6.8 and 6.9. MostFreq and BestBIC approaches both merged a four-elements cluster with its neighbor. Paperrune's GMM detected the largest number of accurate clusters, while the MostFreq produced the most accurate clustering among our GMM implementations. Similarly to the observation made when clustering the HIV set, UCLUST and CD-HIT both failed to cluster this divergent dataset: although UCLUST returned a reasonable number of 6 clusters when the identity parameter was set to 0.5, this result scored a very low ARI when compared to the other approaches.

Let us recall that the "Fast" method is similar to the one used in SpCLUST, except that the K-Means implementation and the random number generator are different, leading to small differences in the results. In the case of $NADH$, the chosen seed used in SpCLUST resulted in a better score than using BestBIC. Indeed, the BestBIC is expected to produce a better clustering than both SpCLUST and Fast. But this may not be respected when the seed of the Fast algorithm is not part of the ones considered in the BestBIC, and when this particular seed leads definitively to a better BIC. This situation can be corrected by increasing the set of possible seeds in the BestBIC approach (100 different seeds are basically considered).

In the $Influenza$ nucleoprotein dataset where the sequences are highly similar, BestBIC scored a perfect ARI, similarly to UCLUST, and CD-HIT. UCLUST and CD-HIT produced equally accurate clusterings, consisting of 3 clusters, when the range of identity thresholds was set to 0.95 or higher. However, the BestBIC approach produced a more balanced clustering consisting of 2 clusters, that is similar to the one produced by UCLUST and CD-HIT for a range of thresholds lower than 0.95. Fast and MostFreq, for their parts, produced only a single cluster. This result is not absurd because the sequences in this dataset are considered very similar for a tool that targets clustering potentially divergent datasets. Applying UCLUST and CD-HIT on this dataset, with identities inferior to 0.88 and 0.91 respectively, also produced a single cluster.

As shown in the previous experiments, traditional tools failed to cluster divergent sequences, while GMM based approaches have been successful. For instance, even though CD-HIT produced a reasonable number of clusters for the HIV dataset (5, with an identity threshold of 0.8, see Figures 6.6 and 6.7), the labels are randomly shuffled, and a reference clustering cannot be deducted to calculate an ARI. Conversely, if UCLUST and CD-HIT succeeded in clustering very similar sequences, like the ones of the $Influenza$ nucleoprotein set, this is too the case for the BestBIC approach: GMM is always interesting, whatever the degree of similarity.

The average Adjusted Rand Index for the clustering of the three sets using our GMM implementation with the BestBIC algorithm is equal to 0.889, followed by SpCLUST and

Paperrune's GMM implementation that scored an average ARI equal to 0.888 and 0.640 respectively. Therefore, on average, the proposed algorithm, relying on the best BIC criterion and using our C++ GMM implementation, outperforms the other evaluated tools, in terms of clustering quality, on the chosen datasets. The algorithm featuring the best BIC, in addition to its good results on the potentially divergent datasets, performs as good as the traditional tools on highly similar sequences. For all these reasons, the algorithms using our C++ GMM implementation are the potential ones to be adopted in SpCLUST-V2, especially if they outperform the Python's original implementation that was adopted in SpCLUST-V1.

A performance comparison between the two versions of SpCLUST has been conducted, by considering the datasets introduced in this article and the 1049 sequences previously used for profiling SpCLUST-V1 [Matar et al., 2019]. This experiment was run over a machine equipped with an i7-6700 3.4GHz processor. Table 6.3 shows the execution times for clustering these data, which includes the computation time of the Eigenmap, using Python's GMM implementation and the new C++ GMM one.

	Python's GMM	Fast	MostFreq	BestBIC
HIV	2,025ms	$<$1ms	4,039ms	1,005ms
NADH	5,046ms	1ms	6,063ms	1,010ms
Influenza	1,008ms	$<$1ms	1,013ms	3ms
1049 sequences	2,280,816ms	53,531ms	82,837ms	60,612ms

Table 6.3: Clustering time using the different implementations.

As can be seen, the new C++ GMM implementation (that is, the Fast approach) achieved up to 42$\times$ speed up when compared to the Python's GMM – and so to SpCLUST-V1 – on the large set of 1049 sequences. The MostFreq and BestBIC also recorded impressive speedups with this dataset, when compared to the Python's GMM. Moreover, the Fast approach achieved much higher levels of speed up on the three smaller datasets where the the Python's GMM performed closely to our most complex approach; the MostFreq. Therefore, the proposed algorithms using this C++ GMM implementation outperforms the Python's GMM of SpCLUST, and it will be adopted in SpCLUST-V2.

6.3.2.2/ SEQUENCES ALIGNMENT IMPACT ON THE CLUSTERING

In this section, the potential influence of the sequence alignment method on the clustering is investigated. The considered sets of sequences have been aligned using four state-of-the-art alignment tools, namely: ClustalX, Biostarts Decipher, MAFFT, and MUSCLE. Then the aligned sequences have been clustered with SpCLUST-V2, BestBIC criterion. Obtained results are presented in Figures 6.11, 6.12, 6.13, 6.14, and 6.15.

Table 6.4 shows the ARI for each clustering, with number of obtained clusters compared

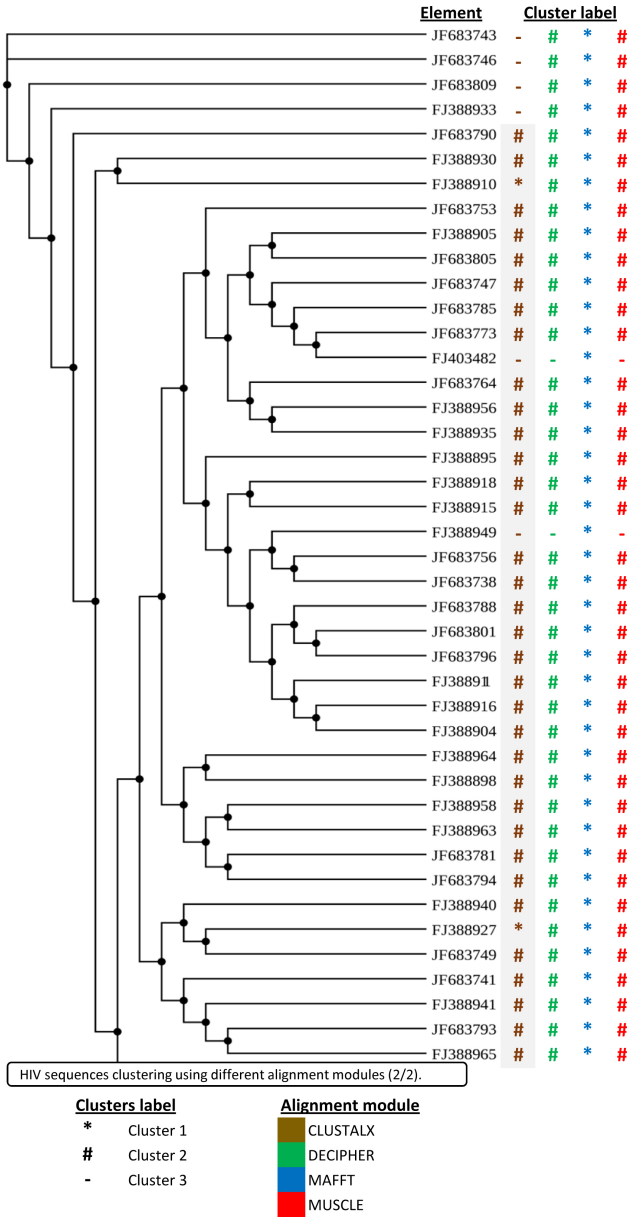

Figure 6.11: HIV sequences clustering using different alignment modules (1/2).

to the ones in the reference.

	HIV			NADH			Influenza		
	Nb. Clusters		ARI	Nb. Clusters		ARI	Nb. Clusters		ARI
	ref.	gen.		ref.	gen.		ref.	gen.	
CLUSTALX	4	3	0.690	4	3	0.955	2	2	1
DECIPHER	3	3	0.759	4	3	0.982	2	2	0.833
MAFFT	1	2	0	2	2	0.960	2	2	1
MUSCLE	3	3	0.828	4	3	0.839	2	2	1

Table 6.4: External clustering validation with regards to the alignment tools.

MUSCLE obtained the largest average ARI (0.889), followed by ClustalX (0.881), Decipher (0.858), and MAFFT (0.653). MAFFT produced less clusters in the HIV and $NADH$

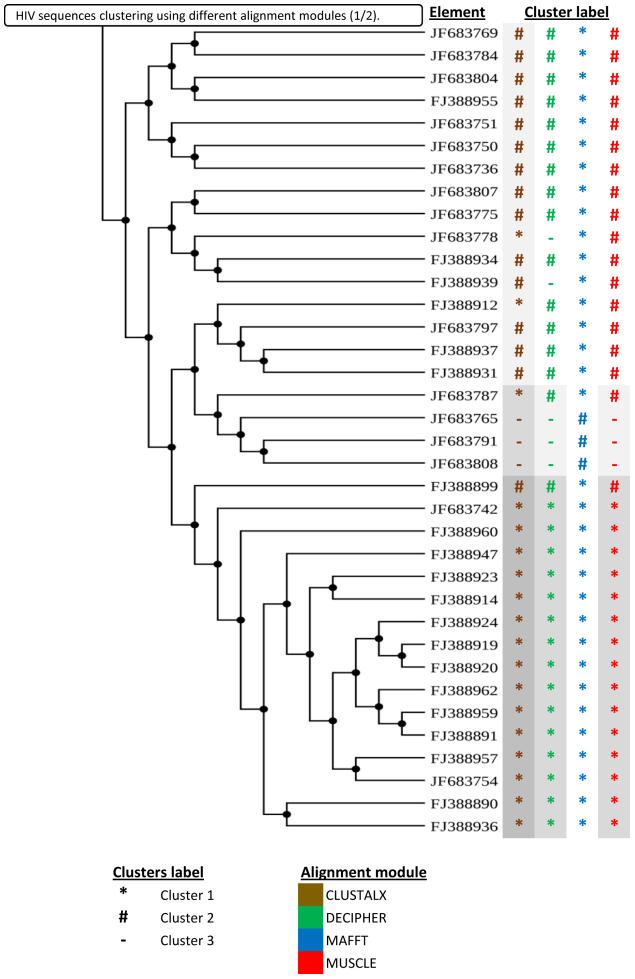

Figure 6.12: *HIV* sequences clustering using different alignment modules (2/2).

set, while Decipher was at the origin of the most parsimonious clustering for the *Influenza* nucleoproteins. Therefore, MUSCLE generally outperforms its competitors. A closer look at the alignment of the *Influenza* proteins set where all the sequences have the same length, shows that DECIPHER was unable to detect SNPs. It considered the mutations between the sequences as insertions and deletions.

The alignment speed is another important aspect for assessing the performance of the alignment tools. To study this, the *HIV* dataset has been selected, as it has the largest size (631 KB). Table 6.5 displays the recorded alignment times on a machine equipped with an Intel core i7-6700 3.4GHz processor and 8GB of RAM[8].

Alignment tool	MUSCLE	CLUSTALX	MAFFT @ 1 thread	MAFFT @ 4 threads	DECIPHER
Alignment duration (seconds)	844	8027	1753	735	115

Table 6.5: Alignment duration for *HIV* sequences using i7-6700 3.4GHz processor.

[8]We have tried to evaluate those tools on a larger dataset of 7MB, but only Decipher was able to perform the alignment, while the other tools required more than 8GB of memory.

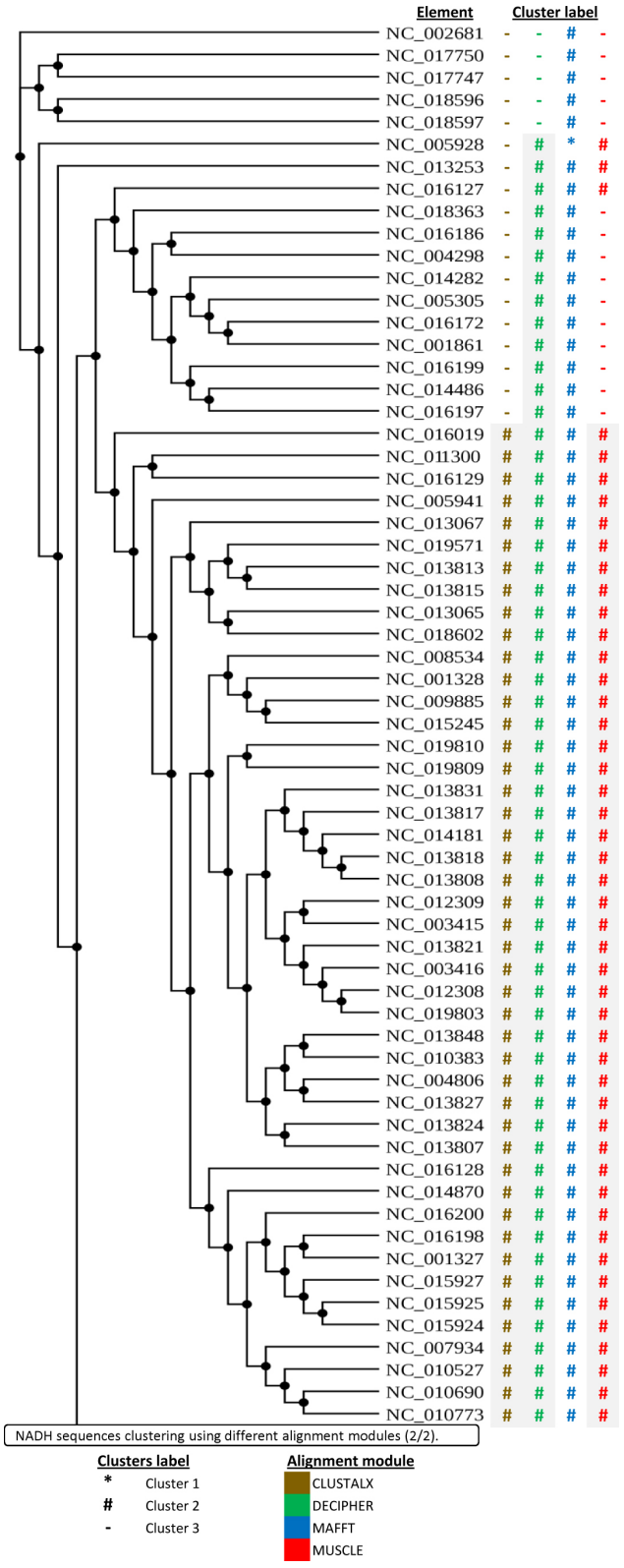

Figure 6.13: *NADH* sequences clustering using different alignment modules (1/2).

As can be seen, Decipher is the fastest alignment tool. It is followed by MUSCLE when using a single-threaded process; but MAFFT outperforms it with 4 threads. The use of

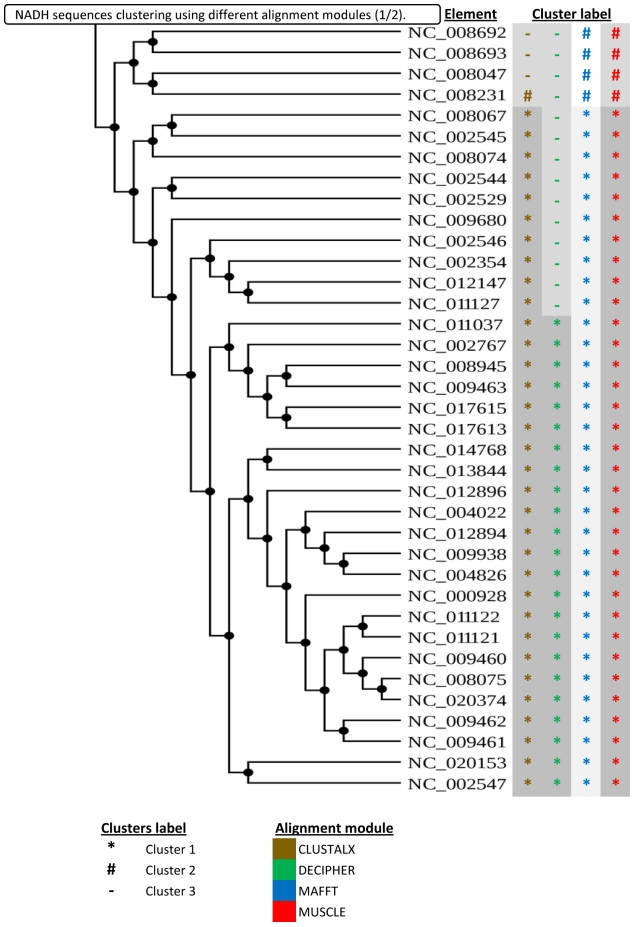

Figure 6.14: $NADH$ sequences clustering using different alignment modules (2/2).

8 threads has not been considered in the evaluation of MAFFT, as it required more than 8GB of memory. Finally, ClustalX is by far the slowest software. To sum up, although Decipher is the fastest tool, its use is problematic, as:

1. the clusters it provides lead to a lower average ARI than the other methods;
2. this is function from an R-language library, i.e., not a standalone executable easily integrated into another C++ package.

Concerning MAFFT, on the one hand, a standalone package exists for both Linux and Windows, and on the other hand, the clustering was fast. However, the clusters it produced scored the lowest average ARI and its package is large, with a size exceeding 60MB, while MUSCLE scored the highest average ARI and consists of a single and small executable. Therefore, MUSCLE was kept as the alignment tool embedded in SpCLUST-V2 and used when no alignment is provided. But the following clarifications can be deduced from the experiments presented:

- MUSCLE is the best suited for small datasets and it delivers the most accurate

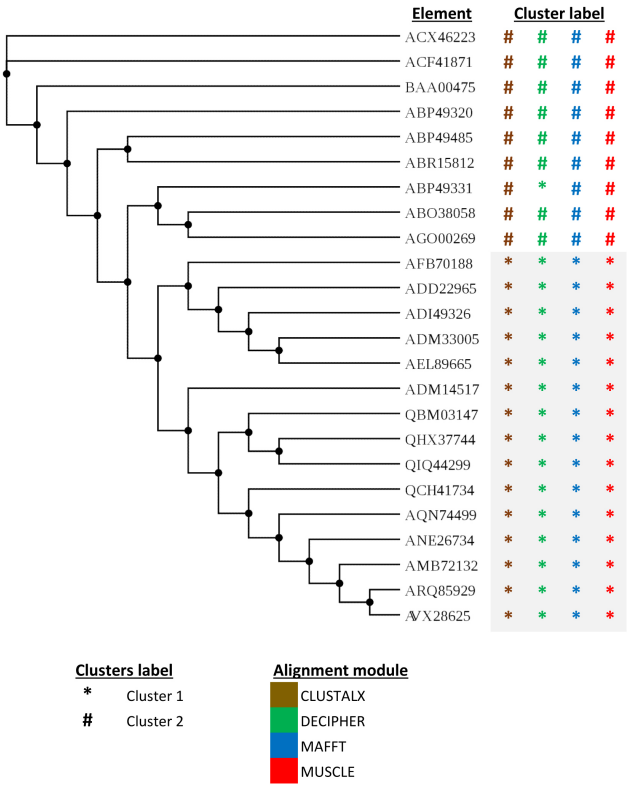

Figure 6.15: *Influenza* sequences clustering using different alignment modules.

results;

- MAFFT with a multi-threaded execution is faster for medium-sized datasets, but it potentially produces less clusters;
- Decipher is the best for large datasets, requiring significantly less resources.

6.3.2.3/ IMPACT OF THE AFFINITY MATRIX

As introduced previously, Non-normalized Laplacian, Modularity, and Bethe Hessian types of affinity matrices have been added to SpCLUST-V2. They are compared in this section, in which BestBIC is used as clustering method. Obtained results are provided in Figures 6.16, 6.17, 6.18, 6.19, 6.20, with a similar external validation than in the previous experiments.

Table 6.6 shows the ARIs indicating, on the one hand, that the Non-normalized Laplacian matrix produced a bad clustering quality, in the case of the *HIV* dataset, with a very low ARI equal to 0.057 and only a few clusters. On the other hand, the use of the Modularity and Bethe Hessian matrices produced the best clustering for this dataset, with an ARI equal to 0.831. The Normalized Lapacian matrix also produced a good clustering, scoring an ARI equal to 0.828.

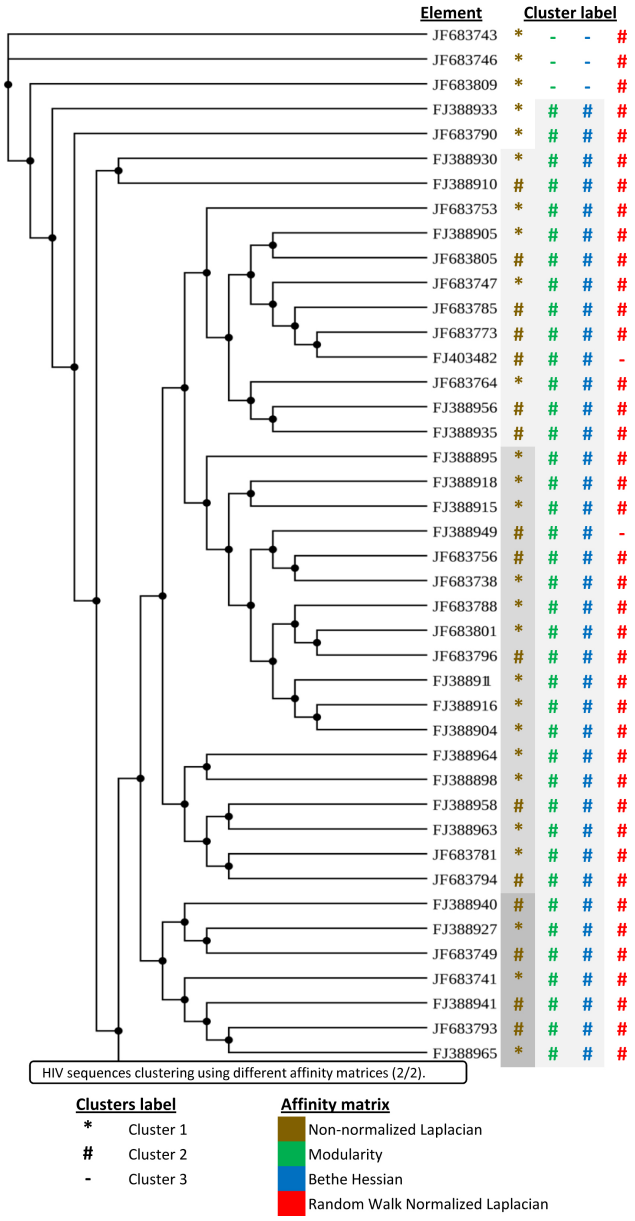

Figure 6.16: *HIV* sequences clustering using different affinity matrices (1/2).

	HIV			NADH			Influenza		
	Nb. Clusters			Nb. Clusters			Nb. Clusters		
	ref.	gen.	ARI	ref.	gen.	ARI	ref.	gen.	ARI
Non-normalized Laplacian	7	2	0.057	1	1	-	2	2	1
Modularity	4	3	0.831	4	3	0.968	3	3	0.857
Bethe Hessian	4	3	0.831	4	3	0.968	2	2	1
Normalized Laplacian	3	3	0.828	4	3	0.839	2	2	1

Table 6.6: Adjusted Rand Index with regards to the used affinity matrix.

In the *NADH* case, the best ARI (0.968) were obtained with the Modularity and Bethe Hessian matrices. The Normalized Laplacian scored 0.839 while the Non-normalized Laplacian produced a single cluster. This last one caused a failure in detecting the different communities of this set. Finally, the lowest (yet good) ARI was obtained with the

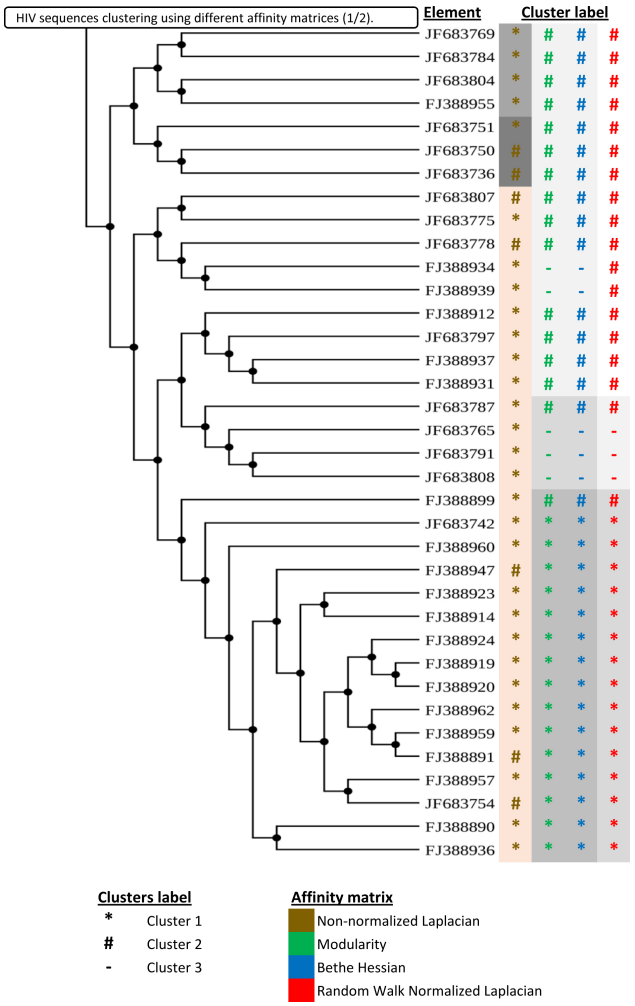

Figure 6.17: HIV sequences clustering using different affinity matrices (2/2).

Modularity matrix in the $Influenza$ nucleoprotein set. The other matrices all produced highly similar clusterings that scored a perfect ARI. The clustering in the Modularity case also had a larger number of clusters than the others, with only a single misplaced element, leading to the detection of more hidden communities in this case.

The results shown in Table 6.6 are consistent with Figures 6.16, 6.17, 6.18, 6.19, and 6.20. As can be seen in Figures 6.16 and 6.17, the Non-normalized Laplacian clustering identifies seven noisy clusters containing 75 elements, and another small cluster of only 3 elements. Conversely, the clusterings produced by using the Modularity and Bethe Hessian matrices identify 4 clusters each, and are similar. The use of the Normalized Laplacian leads to three clusters with a maximum of 2 wrongly labeled elements in the largest one.

Moreover, by using the Non-normalized Laplacian, highly divergent elements of the $NADH$ dataset were grouped into one cluster, see Figures 6.18 and 6.19. Therefore, the use of this matrix is not suitable for too divergent sequences. The cases of Modularity

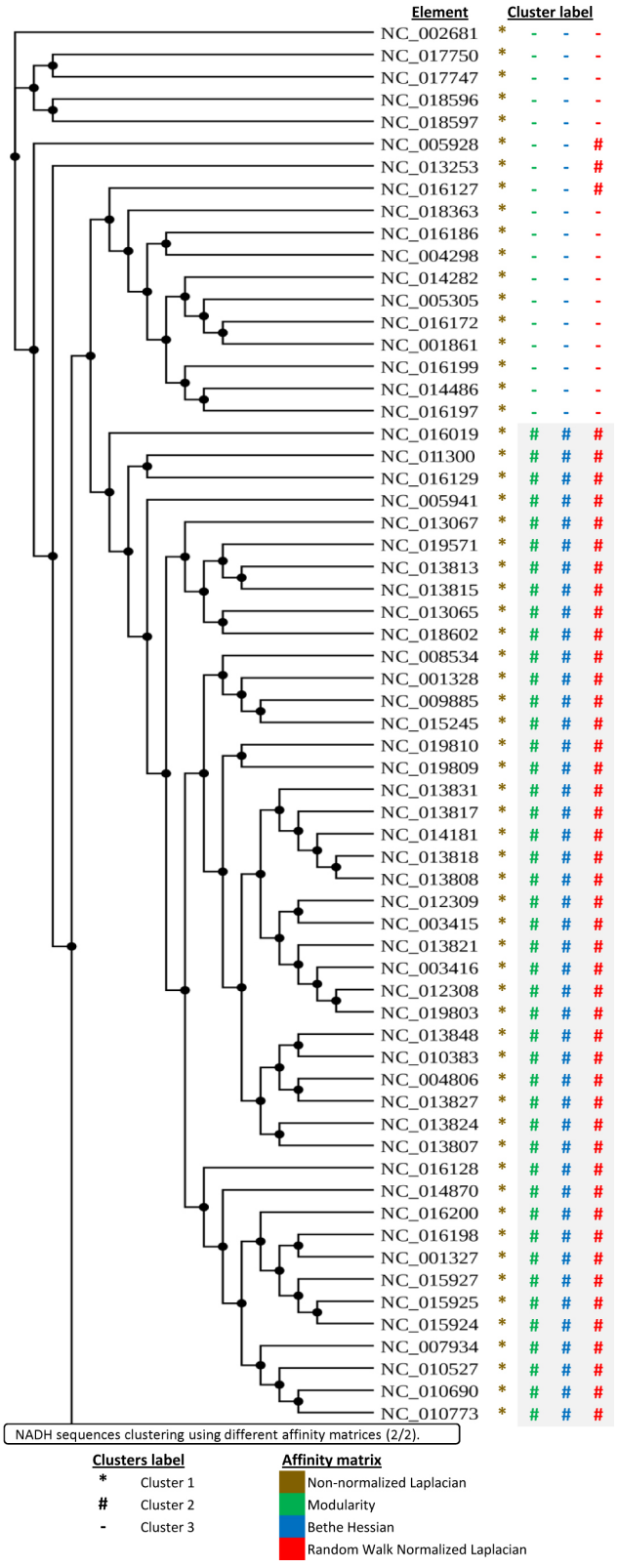

Figure 6.18: *NADH* sequences clustering using different affinity matrices (1/2).

and Bethe Hessian lead to pure clusters (all the elements of the cluster hold a similar label) and thus outperform the Normalized Laplacian in this case. Finally, Figure 6.20

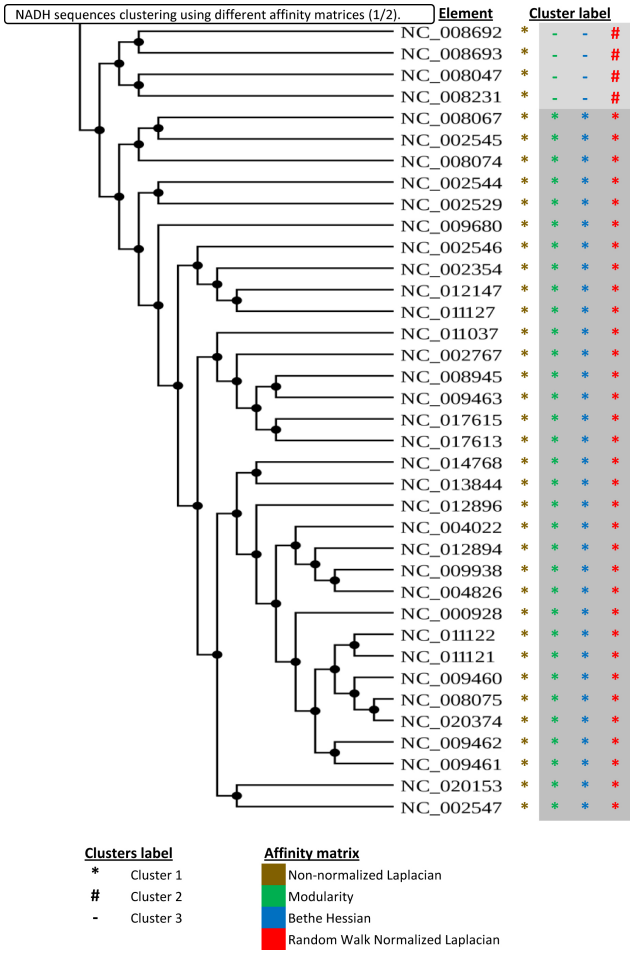

Figure 6.19: $NADH$ sequences clustering using different affinity matrices (2/2).

shows that the use of any of the four matrices returns a good quality clustering. Indeed, the clustering, produced by using the Modularity, contained 3 clusters versus 2 for the others. We recall that these 3 clusters isolated three visually identifiable sub-trees in this figure with a single wrongly clustered sequence: the Modularity matrix allowed a higher detection sensitivity in the case of clustering highly similar data. Conversely, the other types of matrices scored a perfect ARI with a lower detection sensitivity.

To summarize, the Non-normalized Laplacian is only suitable in the case of clustering highly similar sets. The other matrices produce good and similar results in the case of clustering divergent data, and among the latter, the Modularity leads to the detection of a higher number of clusters for highly similar set. Further assessment of SpCLUST-V2 is conducted in the next section.

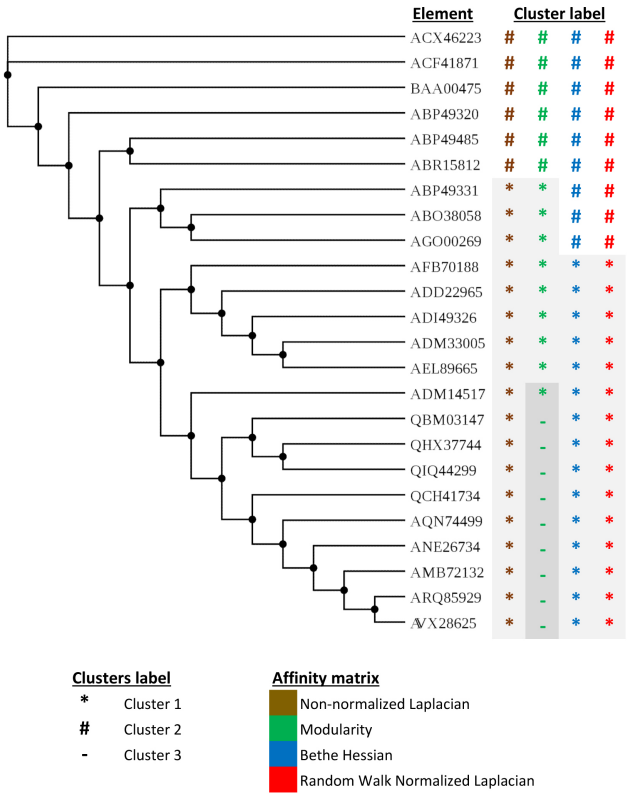

Figure 6.20: *Influenza* sequences clustering using different affinity matrices.

6.3.3/ INTERNAL CLUSTERING VALIDATION FOR SPCLUST-V2

The external clustering validation methods mainly rely on the comparison between two clusterings: the computed clustering and a reference that should be known a priori. We recall that, as described in the experimental protocol, the reference of each dataset was created based on its phylogenetic tree. These trees were generated using PhyML, thus their exactitude remains questionable. In order to provide an independent assessment of the obtained clusterings with SpCLUST-V2, an internal clustering validation is presented in this section. Several internal clustering validation methods and indices were proposed in the literature, and three indices were selected for validating the results of SpCLUST-V2: Silhouette, Davies-Bouldin, and Calinski-Harabasz.

These three internal validation indices are applied to the clusterings obtained from SpCLUST-V2 to evaluate two choices: the clustering method (Fast, MostFreq, and Best-BIC) and the affinity matrix (Non-normalized Laplacian, Normalized Laplacian, Modularity, and Bethe Hessian). Since the use of the Normalized Laplacian matrix scored the best ARI for two out of three sets (Table 6.6), this type of affinity matrix is selected for the internal validation of the proposed clustering algorithms. Table 6.7 shows the recorded internal validation indices for the results of SpCLUST-V2.

Similarly to what has been previously applied for the ARI computation, any clustering

dataset	HIV			NADH			Influenza		
Clustering criterion	Fast	MostFreq	BestBIC	Fast	MostFreq	BestBIC	Fast	MostFreq	BestBIC
Silhouette	0.625	0.674	0.654	0.330	0.423	0.420	-	-	0.355
Davies-Bouldin	0.691	0.810	2.207	1.307	1.227	1.031	-	-	1.356
Calinski-Harabasz	30.720	24.261	36.447	39.241	11.431	32.472	-	-	19.790

Table 6.7: Internal clustering validation with regards to the algorithms.

containing a single cluster is omitted, which applies to the clustering of the $Influenza$ nucleoproteins using both Fast and MostFreq criteria. For the remaining clusterings over the three sets, SpCLUST-V2 scored a $Silhouette$ ranging between 0.330 and 0.674. These values fall in the upper third interval of this index range. The DB indices for the same clusterings were mostly small and ranged between 0.691 and 2.207, while the CH indices were relatively high and ranged between 11.431 and 39.241.

The calculated indices for the clustering of the HIV set gave conflicting results with equal credibility to the three assessed algorithms. Indeed, the Fast algorithm scored the best (lowest) DB and the worst (lowest) Silhouette, while the MostFreq algorithm scored the best (highest) Silhouette and the worst (lowest) CH, and the BestBIC scored the best (highest) CH and worst (highest) DB. A close inspection to the clusterings, illustrated in Figures 6.6 and 6.7, shows that only 3 to 5 elements out of the 78 sequences are differently labelled by using these three algorithms. Therefore, the clustering results in this case are highly similar. Similarly, for the NADH set, the Fast algorithm was favored by the CH index, while the MostFreq was favoured by the Silhouette, and the BestBIC was favored by DB. The high similarity of the results can be examined in Figures 6.8 and 6.9. Finally, we recall that the Fast and MostFreq algorithms produced a single cluster for the Influenza set, therefore the indices were computed for the BestBIC algorithm only.

By disqualifying the algorithms that produced clusterings containing a single cluster, the BestBIC algorithm is the only one that returned valid clusterings (by detecting sub-communities) for the three datasets. So this algorithm was selected for the internal validation of the results with regards to the chosen affinity matrix. Table 6.8 shows the results of these experiments where the Non-normalized Laplacian, Normalized Laplacian, Modularity, and Bethe Hessian matrices are referred to as “UL”, “NL”, “Mod”, and “BH” respectively. The single cluster result for NADH using Non-normalized Laplacian matrix was discarded.

dataset	HIV				NADH				Influenza			
Affinity matrix	UL	NL	Mod	BH	UL	NL	Mod	BH	UL	NL	Mod	BH
Silhouette	0.505	0.654	0.592	0.555	-	0.420	0.390	0.575	0.654	0.355	0.555	0.778
Davies-Bouldin	1.514	2.207	0.999	1.073	-	1.031	1.492	0.905	0.816	1.356	0.689	0.253
Calinski-Harabasz	0	36.447	3.965	4.180	-	32.472	30.493	63.110	17.685	19.790	9.081	148.491

Table 6.8: Internal clustering validation with regards to the affinity matrix type.

The internal validation indices confirmed the validity of the previous assessments based

on the external ones. More precisely, we have previously established that the Normalized Laplacian matrix gave the best clusterings for the HIV while the Modularity or the Bethe Hessian matrices returned the best ones for the NADH sequences. The Normalized Laplacian or the Bethe Hessian matrices returned the best clustering for the Influenza nucleoprotein sequences. Conversely, the clustering of the HIV sequences using the Non-normalized Laplacian was the worst one. According to the three internal validation indices, the clustering of NADH and Influenza datasets using Bethe Hessian, got the best scores among the other clusterings. The clustering of the Influenza proteins using the Normalized Laplacian and the Non-normalized Laplacian scored the second and third best CH index. The HIV clustering, using the Normalized Laplacian, obtained the best scores for two out of the three indices, while it got the worst scores for two out of the three indices when using the Non-normalized Laplacian.

The results presented in this section provide further validity for the interpretations that were made following the external validation process. Indeed, the calculated internal validation indices mostly reflected the clustering validity similarly to the external validation ones. The capability of SpCLUST-V2, in dealing with real-life scenarios, is investigated in the next section.

6.3.4/ CLUSTERING HETEROGENEOUS DATASETS

In the previous experiments, all the three datasets contained different sequences of the same pathogen or gene. In this last experiment, we evaluate the capability of SpCLUST-V2 in clustering the last four heterogeneous datasets consisting of genomes belonging to different pathogens affecting either the same or different regions of the human body, and simulating the case of occurrence of horizontal genes transfer. SpCLUST-V2 was not provided with any a priori knowledge on the number and the characteristics of the pathogens included in the datasets. The BestBIC algorithm of SpCLUST-V2, that detected the highest number of clusters in the previous sets of experiments, was used in this experiment. Since the fifth and seventh sets consist of complete medium size genomes, MAFFT was used for the alignment of these sequences. The Normalized Laplacian (NL), Bethe Hessian (BH), and Modularity (Mod) affinity matrices, embedded in SpCLUST-V2 were also used in this set of experiments. To compare SpCLUST-V2 to its state of the art competitors, we tried to cluster the same four datasets with CD-HIT and UCLUST. Table 6.9 shows the number of expected clusters (known a priori), the number of generated clusters, and the Adjusted Rand Index calculated for each produced clustering.

A closer look to Table 6.9 shows that SpCLUST-V2 accurately clustered all the datasets when using any affinity matrix. The involved genomes were grouped as a single cluster per pathogen type. With CD-HIT, the minimum supported similarity threshold was set to

	dataset 4			dataset 5			dataset 6			dataset 7		
	Nb. Clusters			Nb. Clusters			Nb. Clusters			Nb. Clusters		
	exp.	gen.	ARI	exp.	gen.	ARI	exp.	gen.	ARI	exp.	gen.	ARI
SpCLUST-V2 (NL)	2	2	1	3	3	1	3	3	1	4	4	1
SpCLUST-V2 (BH)	2	2	1	3	3	1	3	3	1	4	4	1
SpCLUST-V2 (Mod)	2	2	1	3	3	1	3	3	1	4	4	1
CD-HIT (id=0.8)	2	4	0.479	3	6	0.570	3	6	0.587	4	8	0.627
UCLUST (id=0.5 till 0.9)	2	2	1	3	3	1	3	11	0.775	4	12	0.816

Table 6.9: Adjusted Rand Index with regards to the used clustering tool.

0.8 and it produced clusterings where the sequences of each pathogen were split into 2 clusters. UCLUST was evaluated with different identity thresholds, starting with an identity threshold equal to 0.9 and at each next experiment it was decremented by 0.1. However, the best results, were obtained with the threshold range of 0.5 till 0.9, which are displayed in Table 6.9. At this threshold range, UCLUST successfully grouped the sequences of the fourth and fifth datasets. Conversely, the HIV sequences that are present in the sixth and seventh sets were clustered as singletons. On the other hand, the Influenza sequences were wrongly grouped when using thresholds lower than 0.5: at a threshold of 0.4 three sequences of Influenza D were grouped in the cluster of Influenza A, and at a threshold of 0.3 all the Influenza sequences were clustered together except one sequence that was classified in the SARS-COV cluster. With higher identities (≥ 0.91) the correct clusters were split.

Finally, the effect of the simulated horizontal gene transfer on the phylogenetic signal, in the last set, is illustrated in Figure 6.21. For a better legibility, the sequences were renamed as follows: the names starting with FA refer to the Influenza A sequences and, similarly, those starting with FD, Co, and HIV refer to the Influenza D, SARS-COV, and HIV sequences respectively. The numbers that are separated by hyphens, in the remainder of the name, refer to the parent sequences, e.g., the sequence named $FA1\_3\_5\_7$ is a descendant of $FA00\_1\_3\_5$, that is in its turn a descendant of $FA0000\_1\_3$, etc... Each kind of the highlighted sequences, that received such gene transfer, and marked by an additional M in their names, were positioned as leafs in the same subtree. This positioning is not absurd because the transferred genes are identical and are inserted in the same positions, therefore they increase the similarity between the affected sequences. Conversely, the phylogenetic tree clearly does not position the sequences as expected: four large subtrees where each one of them consisting of just one kind of sequences, HIV, SARS-COV, Influenza A or Influenza D, while the Influenza subtrees are close to each other or have the same parent node. In the obtained phylogenetic tree, the Influenza D sequences are separated into three distant subtrees while the Influenza A sequences are in the same subtree as the HIV and SARS-COV sequences which could lead to believe that Influenza A sequences are closer to the HIV and SARS-COV sequences than to the Influenza D sequences. Therefore, accurately identifying the correct clusters and

their number became visually impossible while using this tree. This fact demonstrates that our tool successfully handles the cases of horizontal gene transfer in its clustering, conversely from the use of phylogenetic trees. Moreover, building the phylogenetic tree[9] of Figure 6.21 took 9 hours and 18 minutes, compared to a few seconds for clustering the concerned set with SpCLUST-V2.

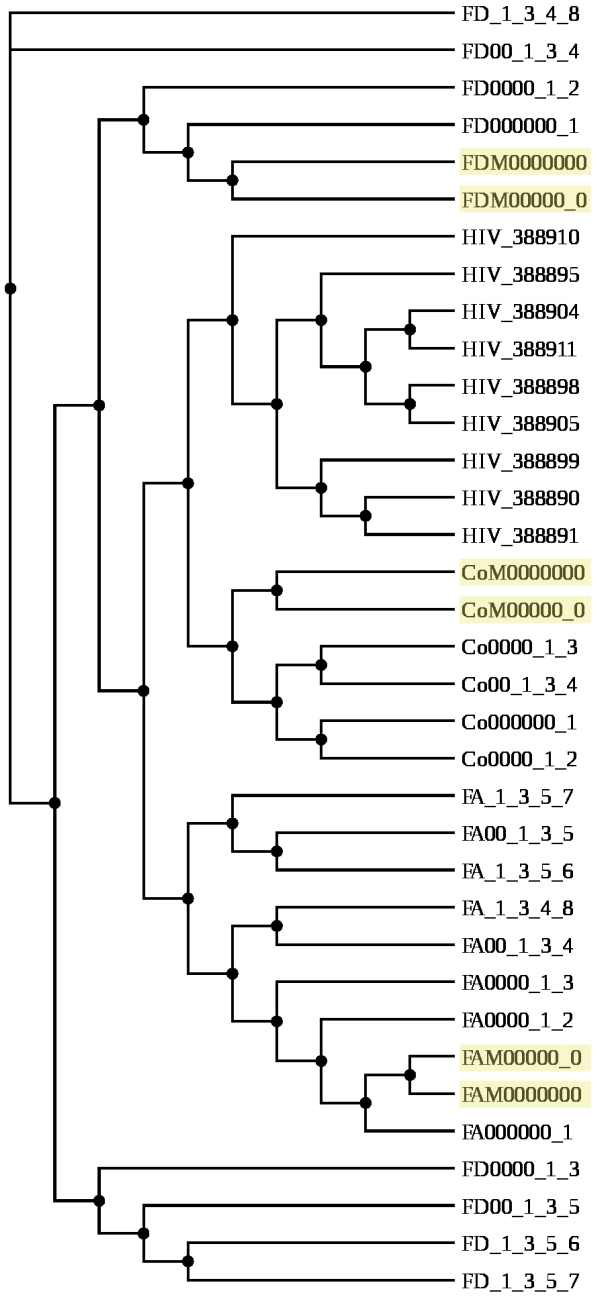

Figure 6.21: Phylogenetic tree of the last hybrid set.

This last assessment demonstrated that SpCLUST-V2 is more capable than the state of the art tools in clustering datasets containing different types of genomes. The experi-

[9]by using PhyML with the same parameters that were used for generating the previous trees

ments also showed that all the embedded affinity matrices are well suited in the case of clustering datasets containing different types of sequences, including those containing sequences that are subject to horizontal gene transfers.

6.4/ CONCLUSION

In this chapter, SpCLUST-V2, an efficient clustering package for both highly similar and divergent biological sequences, is proposed. This package presents major enhancements when compared to its predecessor, SpCLUST presented in Chapter 5. It relies on new algorithms and a new Gaussian Mixture Model (GMM) implementation in C++. The use of the new GMM implementation greatly enhances the performance of the proposed package, when compared with its predecessor that used a Python GMM implementation. A performance comparison for the clustering phase, between the previous package and the new one, shows a speed-up ranging from 9.78× to 15×.

The effects of using four different alignment tools were discussed. The alignment speed of these tools, along with their capacity in handling large sequences, was briefly tackled. Moreover, three added criteria enable the user to choose between a fast single random seed run, the most frequent clustering over several iterations with different seeds, or the clustering scoring the best BIC from a user-defined number of iterations. For a better performance in the last criterion choice, it is possible to stop iterating if no improvement is detected after a certain user-defined number of consecutive iterations. Three additional affinity matrices are also available in SpCLUST-V2.

A comparative study between SpCLUST-V2, SpCLUST [Matar et al., 2019], UCLUST [Edgar, 2010], and CD-HIT [Li et al., 2006] was conducted over three different sets of real genomic and protein sequences. In contrast with most of the state of the art tools, SpCLUST-V2 aims for a fast spectral clustering of datasets, regardless of the level of similarity or divergence of their elements. Similarly to its predecessor, an a priori knowledge of the similarity threshold or the number of clusters is not required. The results of the experiments show that SpCLUST-V2 produced similar or better clusters contents than the other tested tools for both highly similar and highly divergent datasets, in addition to successfully handling horizontal genes transfer. This comparison was based on an external clustering validation index. Three additional internal clustering validation indices further supported the results validity. The ability of SpCLUST-V2 in clustering datasets containing different types of sequences was also proved.

In the next chapter, an additional improvement in the computation of the pairwise similarities is achieved. Moreover, additional clustering techniques will be exploited in the biological sequences clustering field. Finally, a novel clustering algorithm is introduced.

7

NOVEL CLUSTERING APPROACHES: A COMPARATIVE STUDY

7.1/ INTRODUCTION

The tools presented in [Bruneau et al., 2018] and the previous chapters of this thesis, use GMMs (Gaussian Mixture Models) for performing an unsupervised clustering of biological sequences. Although they returned good quality clusterings for highly divergent sequences and overlapping clusters, they are not adapted to process large datasets because they include two computationally intensive tasks in their pipeline: i- the pairwise affinity computation of the sequences, ii- the clustering using the GMM itself. We recall that this issue has been addressed in Chapters 5 and 6, by using a MPI (Message Passing Interface) parallel computation scheme and re-coding some parts in a lower level programming language in order to accelerate the process, but the computation time of the affinity matrix remained problematic.

Moreover, to our knowledge, many clustering techniques among the ones presented in our state of the art, are being used in other fields and have never been experimented in the biological sequences clustering field. These techniques are DBSCAN [Khan et al., 2014], HDBSCAN [Campello et al., 2013], and a recent technique that relies on connectivity patterns [Benson et al., 2016], called MOTIFS. A common advantage for these techniques is there ability of detecting random shaped clusters, which is the case for clustering biological sequences in an n-dimensional space by using the spectrum (eigenvalues). Therefore, they present potential solutions for clustering biological sequences.

The contributions presented in this chapter are thus threefold: 1- assuring an enhanced affinity computation time compared to the packages introduced in the previous chapters, 2- presenting a qualitative study for the results of three additional and not experimented clustering algorithms in the field of biological sequences, 3- proposing and assessing a

novel and lightweight clustering algorithm. The experimented methods and algorithms were integrated to an improved and full featured clustering package[1].

The remainder of this chapter is organized as follows. In Section 7.2, the datasets that are used in these experiments are presented. Section 7.3 presents the achieved improvement in the alignment and affinity computation time. Section 7.4 contains a qualitative study, of some existing clustering techniques, in the field of biological sequences clustering. The novel CHAINS clustering technique is introduced and evaluated in Section 7.5. An additional numerical evaluation and comparison, between the results of the explored clustering techniques and those of some existing biological sequences' clustering tools, is presented in Section 7.6. Finally, Section 7.7 concludes this chapter.

7.2/ THE EXPERIMENTAL DATASETS

The datasets that were used for the assessment of the clustering quality, are composed of some of the following subsets of sequences:

- 4 genomic sequences, belonging to the NADH dehydrogenase 3 (ND3) mitochondrial gene.
- 48 complete genomes of SARS-COV2 collected in the UK and belonging to the B.1.1.7 lineage.
- 50 complete genomes of SARS-COV2 collected in South Africa and belonging to the B.1.351 lineage.
- 21 complete genomes of SARS-COV2 collected in Europe and belonging to the B.1.525 lineage.
- 8 spike protein sequences extracted from SARS-COV2 samples belonging to the B.1.177 lineage.
- 3 spike protein sequences extracted from SARS-COV2 samples belonging to the B.1.1 lineage.
- 9 spike protein sequences extracted from SARS-COV2 samples belonging to the B.1.1.7 lineage.
- 5 spike protein sequences extracted from SARS-COV samples.
- And finally, 3 spike protein sequences extracted from MERS samples.

[1] The source code is available on https://github.com/johnymatar/SpCLUST-Global

A detailed list showing the sourcing organisms of the sequences, their accession ID, and the collection date of samples can be found in Appendix V.

Using the NADH genomic sequences, 4 simulated datasets were generated as follows: 2 datasets of 1000 sequences each and 2 larger data sets of 2004 sequences each, following the 2 mutation scenarios presented in Table 7.1. These scenarios involve the following criteria:

1. The mutation rate with regards to the number of bases in the considered sequence.
2. The rate in which the possibility of random gaps can occur.
3. The maximum size of a produced gap.
4. The maximum number of random insertions.
5. The maximum size (number of bases) in an insertion.
6. The number of generations produced during the simulated mutations, i.e. the depth of the produced tree.

Scenario	Mut. rate	Gaps rate	Max. gap size	Max. nb. ins.	Max. ins. size	Nb. gens.
S1	5%	2%	10	3	10	3
S2	10%	5%	20	10	20	5

Table 7.1: Scenarios for simulated mutations.

The produced datasets, using the previously described scenarios, have 2 different sizes and 2 different divergence rates. The data sets produced according to the second scenario $S2$, presented in Table 7.1, naturally have a higher rate of divergence compared to those produced according to $S1$. The 1000-sequences dataset, generated according to $S1$, will be referred to as $NADH-S1$ while the 2004-sequences dataset, generated using the same scenario, will be referred to as $NADH-S1L$. Similarly, the datasets generated using $S2$ will be named $NADH-S2$ and $NADH-S2L$. Since all these datasets are generated from 4 source sequences, they should all produce a clustering of 4 clusters, containing each the descendants of a same source sequence, with either 250 or 501 elements per cluster.

Four additional real datasets were formed as the following:

- A dataset called $COV2-Vars$ that contains the 109 collected SARS-COV2 genomes. This dataset has 3 clusters of the different collected variants (lineages) of SARS-COV2.
- A dataset called $COV2-Spike-UK$ where the 20 spike protein sequences, extracted from the SARS-COV2 samples collected in the UK, are grouped together. This

dataset should also be divided into 3 clusters, where each cluster consists of the spike proteins of a different lineage of the virus.

- $COV-Spike-4$ is a dataset that contains the sequences of $COV2-Spike-UK$ in addition to the 5 collected spike protein sequences of the SARS-COV virus. A typical clustering of this set should contain 4 clusters, while a less sensitive clustering could produce just 2 clusters containing either the sequences belonging to COV2 or COV.

- Finally, $COV-Spike-5$ is a dataset that contains the sequences of $COV-Spike-4$ in addition to the 3 collected spike protein sequences of the MERS virus. Similarly to the interpretation of the previous dataset, this set should be either ideally partitioned into 5 clusters or a less sensitive clustering could produce just 3 clusters.

Data set	Nb. seqs.	Avg. length	Max. simil.	Min. simil.	Avg. simil.	Nb. clusters
NADH-S1	1000	380	0.9096	0.3660	0.5295	4
NADH-S1L	2004	379	0.9571	0.4720	0.5906	4
NADH-S2	1000	584	0.9173	0.3851	0.5302	4
NADH-S2L	2004	586	0.9126	0.3663	0.5285	4
COV2-Vars	119	29780	0.9999	0.9769	0.9916	3
COV2-Spike-UK	20	1271	0.9992	0.9096	0.9860	3
COV-Spike-4	25	1268	0.9992	0.7059	0.9123	4
COV-Spike-5	28	1277	0.9992	0.3366	0.8040	5

Table 7.2: Properties of the datasets.

The properties of the simulated and the real datasets are presented in Table 7.2. In addition to the number of sequences in each dataset and its expected number of clusters, the average length of the sequences, and the maximum, minimum, and average similarity, among the sequences of each set, are also shown in Table 7.2. The difference in the average length, between the datasets generated using the simulation scenarios $S1$ and $S2$, clearly shows the higher rate of random insertions in the sequences produced by $S2$. These data sets will be used to evaluate the clustering accuracy resulting from the improvements and techniques that are introduced in the next sections.

7.3/ IMPROVING THE COMPUTATION SPEED OF THE SIMILARITY MATRIX

In the previous GMM-based clustering tools, namely the GCLUST, SpCLUST, and SpCLUST-V2 packages, the pairwise similarities are computed by comparing pairs of the aligned sequences. A global alignment is performed before the computation of these similarities and MUSCLE [Edgar, 2004] is used by default in these three packages. However, in [Matar et al., 2019] it was shown that MAFFT [Katoh et al., 2013] outperforms

MUSCLE in some cases, in terms of alignment speed and quality of the produced clustering when using its global alignment. Moreover, in [Matar et al., 2021], aligning a set of 78 genomic sequences, having a length of about 1000 bases each, using either MUSCLE or MAFFT, took between 735 and 844 seconds. In order to avoid realigning an already aligned dataset, the ability of providing aligned sequences as input was added to SpCLUST-V2.

Since a good alignment of the sequences remains a fundamental and unavoidable step for the analysis of the biological sequences and the calculation of an accurate pairwise similarity between them, accelerating this process while preserving the quality of the resulting alignment is still the subject of ongoing research. A recent algorithm for pairwise alignment that uses Needleman-Wunsch and is implemented in c/c++ in the EDLIB library, was proposed in [Šošić et al., 2017]. It was shown in [Šošić et al., 2017] that this algorithm has an optimal memory usage and is able to handle very large sequences while outperforming the state of the art libraries. Therefore, we propose using this library to accelerate the computation time of the sequence alignment in SpCLUST.

In order to evaluate the alignment speed and quality of the EDLIB library, the largest datasets, in terms of number of sequences or length of the sequences, were selected. $NADH-S1L$ and $NADH-S2L$ fulfill the first criterion and $COV2-Vars$ the second. In the SpCLUST version that uses EDLIB, the pairwise similarity of the aligned sequences is computed along with the alignment of each pair of sequences as follows:

$$1-\frac{Levenshtein\ distance}{length\ of\ the\ aligned\ sequences}.$$

For the sake of comparison, the alignment of the selected data sets was also generated using the state of the art tools, MUSCLE and MAFFT, that were evaluated as being the best among others in [Matar et al., 2021]. The required time for computing the similarity matrix, by using the aligned datasets, was also recorded. To ensure a fair comparison, all the experiments were conducted on the same machine equipped with an Intel i7-6700 - 3.4GHz Quad-Core (8 Threads) processor and 8GB of RAM. Table 7.3 displays the parameters that were used for the alignment and the recorded observations that include the recorded times for aligning the sequences and calculating the similarity matrix, in addition to the average and maximum lengths of the aligned sequences that were produced.

In the case of MUSCLE, the limit in the iterations number, imposed by the large number of sequences or the large size of the sequences, caused a potential deterioration in the quality of the alignment. A closer look at the aligned sequences of the simulated $NADH$ datasets, reveals a significant increase in the size of these sequences when compared to the size of the original ones. Moreover, the smaller the number of iterations is set, the larger the size of the aligned sequences becomes: indeed, a two-iterations alignment for the $NADH-S1L$ data set caused a 3.96 times (7528 vs. 1900) larger aligned sequences

Data set	Alig. tool	Alig. parameters	Alig. + Simil. calc. time (s)	Avg. length	Max. length
NADH-S1L	MUSCLE	2 iterations	102 + 35361 = 35463	7528	7528
NADH-S1L	MUSCLE	4 iterations	11200 + 2105 = 13305	1900	1900
NADH-S1L	MAFFT	Auto / 1 thread	27 + 19680 = 19707	5758	5758
NADH-S1L	MAFFT	Auto / 8 threads	14 + 19680 = 19694	5758	5758
NADH-S1L	EDLIB	NW / 1 process	423	420	475
NADH-S1L	EDLIB	NW / 8 processes	86	420	475
NADH-S2L	MUSCLE	2 iterations	235 + 137750 = 137985	15206	15206
NADH-S2L	MAFFT	Auto / 1 thread	79 + 192490 = 192569	17841	17841
NADH-S2L	MAFFT	Auto / 8 threads	52 + 192490 = 192542	17841	17841
NADH-S2L	EDLIB	NW / 1 process	805	675	934
NADH-S2L	EDLIB	NW / 8 processes	157	675	934
COV2-Vars	MAFFT	Auto / 1 thread	30 + 56 = 86	29893	29893
COV2-Vars	MAFFT	Auto / 8 threads	16 + 56 = 72	29893	29893
COV2-Vars	EDLIB	NW / 1 process	248	29805	29883
COV2-Vars	EDLIB	NW / 8 processes	52	29805	29883

Table 7.3: Observation of the alignments

compared to a four-iterations alignment. This could be possibly caused by an increased number of identification of false-insertions and false-gaps instead of identifying mutations. In addition to what preceded, while using MUSCLE, the alignment fails if the number of iterations is set respectively to greater than 4 and 2 for the datasets $NADH-S1L$ and $NADH-S2L$. In order to successfully perform more iterations to refine the alignment quality of these sets, MUSCLE requires a higher memory size than the available one. For this same reason, even a single iteration limit could not produce an alignment in the case of the $COV2-Vars$ data set. Finally, a comparison between a two-iterations and a four-iterations alignment attempt on the $NADH-S1L$ dataset with MUSCLE shows that the four-iteration alignment required 109.8 times more time when compared to the two-iterations attempt, besides the significant difference in the size of the produced aligned sequences. This observation further supports the idea of potential quality deterioration when using less iterations and clearly shows how much a good alignment can become time-consuming with MUSCLE.

For these same sets of sequences, the automatically selected algorithm in MAFFT performs faster than MUSCLE, but produces a closely similar size for the aligned sequences. This similarity leads to thinking that the alignment quality is also potentially close to the one produced by MUSCLE. Indeed, a closer look at the recorded values in Table 7.3 shows that, in the alignment results for the $NADH-S1L$ and $NADH-S2L$ datasets, the size of the sequences that were aligned using MAFFT is close to the size of the ones produced by a two-iterations alignment by MUSCLE. As a result, the similarity matrix computation time showed a drastic raise since it scales quadratically[2] with the size of the aligned sequences. MAFFT took 1226.38 more time than EDLIB (192542s vs. 157s) to perform the same computation on the NADH-S2L dataset. 8 threads and/or 8 processes

[2]The order of the computation of similarities is $O(\frac{N^2-N}{2})$, where N is the number of sequences, and each similarity computation time is proportional to the size of the aligned sequences.

were used by both tools for this computation.

Another aspect of comparison is the scaling efficiency of both tools: our parallel algorithm that uses EDLIB (for computing the pairwise alignment and the similarity altogether) and the multi-thread algorithm implemented in MAFFT (for computing the global alignment):

- In the experiments on the $NADH-S1L$ dataset, our implementation scored a time improvement of 79.66% in its parallel computation when compared to its serial one, while MAFFT scored a 48.14% time improvement in similar conditions. These recorded time improvements correspond to a 61.48% and 24.10% strong scaling efficiencies[3] respectively.
- In the experiments on $NADH-S2L$ data set, an 80.49% time improvement and 64.09% strong scaling efficiency was achieved by our implementation compared to a 34.17% time improvement and 18.99% strong scaling efficiency for MAFFT.
- Finally in the case of $COV2-Vars$, our implementation also achieved better time improvement (79.03% vs. 46.66%) and better strong scaling efficiency (59.61% vs. 23.43%).

These values show a significantly better scaling efficiency for our implementation when compared to MAFFT.

Technically, it is normal that EDLIB produces a significantly smaller size of aligned sequences because it does a local pairwise alignment while a global one is done with MUSCLE and MAFFT. For the sake of comparison, a pair of sequences from the $NADH-S2L$ dataset was randomly chosen to compare the size of a local alignment using MUSCLE, MAFFT, and EDLIB with the same parameters that were used for this data set and presented in Table 7.3: the size of the aligned sequences was equal to 910 for MUSCLE, 880 for MAFFT, and 879 for EDLIB. Although the size of the aligned sequences are close in this case, performing a local pairwise alignment with MUSCLE and MAFFT is not practical because the huge number of calls for an external tool will naturally produce a significant time consumption, and this was experimented in [Matar et al., 2019] when an external module was called for the similarity computation of each pair of sequences. Moreover, it was not possible to produce a local pairwise alignment for any pair or $COV2-Vars$ sequences, using MUSCLE, on the same experimentation machine.

A further investigation of the obtained alignments consists in evaluating the quality of the clusterings that are produced using these aligned datasets. The evaluation was done on all of the datasets that were presented in Table 7.2. The obtained number of clusters in the resulting clusterings, along with the computed $purity$ and ARI of these clusterings, are presented in Table 7.4.

[3]The strong scaling efficiency is computed as $\frac{t_1}{t_N * N} * 100\%$ where t_i is the computation time required when

Data set	Alig. method	Alig. param.	Nb. clusters	Purity	ARI
NADH-S1	MUSCLE	Maximum iterations	5	0.589	0.235
NADH-S1	MAFFT	Auto	4	0.750	0.637
NADH-S1	EDLIB	-	3	0.677	0.513
NADH-S1L	MUSCLE	4 iterations	1	-	-
NADH-S1L	MAFFT	Auto	3	0.322	0.009
NADH-S1L	EDLIB	-	2	0.286	0.005
NADH-S2	MUSCLE	2 iterations	3	0.492	0.213
NADH-S2	MAFFT	Auto	4	0.879	0.763
NADH-S2	EDLIB	-	2	0.440	0.181
NADH-S2L	MUSCLE	2 iterations	2	0.290	0.006
NADH-S2L	MAFFT	Auto	1	-	-
NADH-S2L	EDLIB	-	1	-	-
COV2-Vars	MAFFT	Auto	4	0.991	0.820
COV2-Vars	EDLIB	-	5	1	0.718
COV2-Spike-UK	MUSCLE	Maximum iterations	2	0.850	0.744
COV2-Spike-UK	MAFFT	Auto	2	0.850	0.744
COV2-Spike-UK	EDLIB	-	3	0.850	0.691
COV-Spike-4	MUSCLE	Maximum iterations	3	0.840	0.733
COV-Spike-4	MAFFT	Auto	3	0.840	0.733
COV-Spike-4	EDLIB	-	3	0.840	0.733
COV-Spike-5	MUSCLE	Maximum iterations	4	0.892	0.828
COV-Spike-5	MAFFT	Auto	4	0.892	0.828
COV-Spike-5	EDLIB	-	3	0.607	0.355

Table 7.4: Clustering results using the different alignment methods.

In the case of the simulated datasets, where the sequences present a high level of divergence, the produced number of clusters and the computed ARI show that the GMM could hardly produce acceptable results for the two smaller datasets. The use of the sequences that were aligned by MAFFT produced the best results for $NADH-S1$ and $NADH-S2$. The scored ARI also shows that the use of EDLIB for $NADH-S1$ also produces acceptable results, while the remaining results on the simulated datasets score a very low ARI, thus a bad clustering. This ARI also shows that the GMM failed to produce a valid clustering for the larger data sets. Because of the high divergence, and subsequently greater distance, between these sequences, their inferred data points became very dispersed in the plane, while producing random shapes. Therefore, the discovery of their correct clusters became harder for the GMM. The large number of sequences causes even more noise and complexity which leads to a single cluster grouping, by the GMM, as shown in 3 out of the 6 clusterings produced for the large datasets.

Conversely, the clustering results of the real datasets demonstrate the validity of the clusterings produced using either one of the alignment tools. The use of EDLIB allowed the GMM to detect more sub-clusters in the case of $COV2-Vars$ and $COV2-Spike-UK$. Sub-grouping a same cluster is penalized when computing the ARI for this clustering. It led to a lower ARI for EDLIB when compared to the other results of these two datasets. However, this same sub-grouping obtained a perfect $purity$ score for EDLIB in the case of clustering the $COV2-Vars$ sequences. The clusterings of the $COV-Spike-4$ se-

using i processes or threads and N is the number of processes or threads.

quences obtained the same scores with the three tools, while EDLIB scored lower results for the $COV - Spike - 5$ sequences. However, a closer inspection of that last clustering shows that the three species of sequences (SARS-COV2, SARS-COV, and MERS) were correctly separated into three clusters. However, the clustering failed to differentiate the three lineages of SARS-COV2.

Finally, based on the important gain in performance coupled with the ability to handle very large sequences and since there was no significant deterioration in the clustering quality, the implementation of EDLIB was considered as the most interesting alignment solution. EDLIB was also used in the rest of this chapter's experiments.

7.4/ EVALUATING SOME STATE OF THE ART CLUSTERING METHODS

In the present section, the clustering results of the state of the art approaches or algorithms, that were not previously exploited in the field of clustering biological sequences, are evaluated.

7.4.1/ THE MOTIFS APPROACH

This novel clustering method for clustering higher order networks [Benson et al., 2016] computes a kind of affinity matrix via MOTIFS-counting, then uses the EM-GMM for clustering. This algorithm was initially introduced for clustering large graphs with directed links, that represent the relations, among its nodes, and where the MOTIFS are a set of 13 triangular shapes having each a different combination of directions on its sides. Since the relation between the biological sequences is bidirectional, then only two MOTIFS may be used in this case. Figure 7.1 illustrates the different MOTIFS and highlights the usable ones for clustering biological sequences, precisely the M4 and M13.

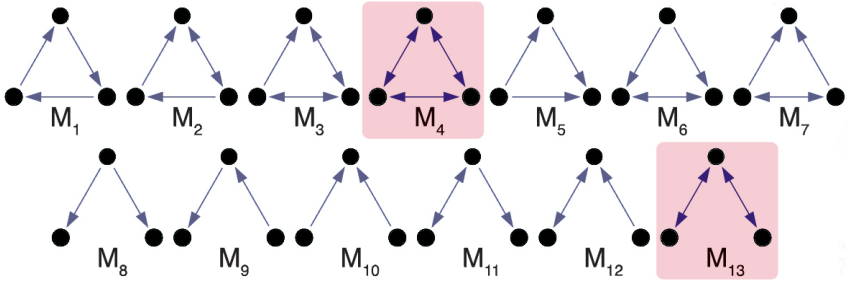

Figure 7.1: MOTIFS.

To cluster biological sequences through this approach it is mandatory to define when

two sequences are considered to be linked. Indeed, there is no pair of sequences that scores a zero-valued similarity. Therefore, a certain similarity threshold should be set and two sequences with a similarity score below this threshold are considered as unrelated. Therefore, in order to set such a threshold, we propose the following experimental techniques:

- $Deltas$ is a semi-manual technique that consists in computing the dissimilarity Δ between each sequence i, and its closest one as follows:

  $$\Delta_i = 1 - \textit{the highest similarity with the sequence } i.$$

  Then, using a user-input threshold $th > 1$, the minimum acceptable similarity to establish a link between the sequence i and another sequence is computed as follows:

  $$minSimil_i = 1 - (\Delta_i * th)$$

  This technique avoids the possibility of keeping links between potentially distant sequences that might be within a certain constant threshold.

- AVG is another simple automatic technique where the average of the computed similarities among the input sequences is calculated. Only the links between sequences having a pairwise similarity greater than the average are kept. This technique can be extended to keeping the similarities higher than the average multiplied by an user-defined threshold.

- Finally, $Auto$ is an iterative technique where at each iteration, a graph is generated as follows: each sequence keeps a certain x percentage of the links with the other sequences. For example, each sequence will keep its links with the $x\%$ of the remaining sequences that score the highest similarity with it. The resulting graph is used to produce a clustering, then the internal clustering validation index, Calinski Harabasz (CH), is computed. Between iterations, the value of x varies from 0.05 to 0.8 with a step equal to 0.05. Finally, the clustering with the highest CH index is selected. This iterative technique is highly compute intensive because the data is clustered at each iteration.

In the experiments, on the eight previously described datasets in Section 7.2, multiple trials were performed for the $Deltas$ technique with different threshold values. The tested thresholds ranged from 1.5 to 9 with a step equal to 0.5. The details of the resulting clusterings that scored the best ARI are presented in Tables 7.5 and 7.6.

An overview of the presented results in Tables 7.5 and 7.6 shows that the use of both MOTIFS (M4 and M13) produces highly similar results. These results are also not very different from the ones produced using the traditional GMM scheme with an alignment

Data set	T. tech	Threshold	Nb. clusters	Purity	ARI
NADH-S1	Deltas	3.0	4	0.445	0.043
NADH-S1	AVG	-	2	0.257	1.7e-4
NADH-S1	Auto	-	3	0.542	0.341
NADH-S1L	Deltas	3.0	5	0.385	0.018
NADH-S1L	AVG	-	3	0.278	0.001
NADH-S1L	Auto	-	3	0.285	0.004
NADH-S2	Deltas	2.5	3	0.297	0.005
NADH-S2	AVG	-	2	0.252	5.2e-6
NADH-S2	Auto	-	3	0.485	0.288
NADH-S2L	Deltas	2.0	4	0.272	8.1e-4
NADH-S2L	AVG	-	2	0.251	6.0e-6
NADH-S2L	Auto	-	2	0.250	0.000
COV2-Vars	Deltas	5.5	3	0.789	0.522
COV2-Vars	AVG	-	3	0.571	0.041
COV2-Vars	Auto	-	4	0.941	0.667
COV2-Spike-UK	Deltas	3.0-6.0	3	0.850	0.691
COV2-Spike-UK	AVG	-	2	0.500	0.005
COV2-Spike-UK	Auto	-	2	0.850	0.744
COV-Spike-4	Deltas	4.0-6.0	3	0.840	0.733
COV-Spike-4	AVG	-	3	0.600	0.285
COV-Spike-4	Auto	-	3	0.720	0.548
COV-Spike-5	Deltas	4.0-6.0	3	0.750	0.654
COV-Spike-5	AVG	-	3	0.607	0.375
COV-Spike-5	Auto	-	3	0.500	0.251

Table 7.5: Clustering results using the $M4$ MOTIF.

Data set	T. tech	Threshold	Nb. clusters	Purity	ARI
NADH-S1	Deltas	3.0	5	0.406	0.027
NADH-S1	AVG	-	3	0.467	0.200
NADH-S1	Auto	-	2	0.252	2.7e-6
NADH-S1L	Deltas	3.0	5	0.385	0.017
NADH-S1L	AVG	-	2	0.318	0.020
NADH-S1L	Auto	-	3	0.284	0.004
NADH-S2	Deltas	2.5	3	0.277	0.002
NADH-S2	AVG	-	2	0.252	5.2e-6
NADH-S2	Auto	-	2	0.484	0.287
NADH-S2L	Deltas	2.5	3	0.262	4.0e-4
NADH-S2L	AVG	-	2	0.251	6.0e-6
NADH-S2L	Auto	-	2	0.250	0.000
COV2-Vars	Deltas	6.0	3	0.789	0.524
COV2-Vars	AVG	-	3	0.571	0.041
COV2-Vars	Auto	-	4	0.932	0.654
COV2-Spike-UK	Deltas	3.0-6.0	3	0.850	0.691
COV2-Spike-UK	AVG	-	2	0.500	0.005
COV2-Spike-UK	Auto	-	2	0.850	0.744
COV-Spike-4	Deltas	4.0-6.0	3	0.840	0.733
COV-Spike-4	AVG	-	2	0.560	0.294
COV-Spike-4	Auto	-	3	0.720	0.548
COV-Spike-5	Deltas	4.0-6.0	4	0.785	0.617
COV-Spike-5	AVG	-	3	0.607	0.375
COV-Spike-5	Auto	-	3	0.535	0.330

Table 7.6: Clustering results using the $M13$ MOTIF.

by EDLIB and presented in Table 7.4. In the following, a discussion of the experiment's results:

- The best clustering for the $NADH-S1$ dataset scored an ARI of 0.341 and a $purity$ of 0.542 when using the $Auto$ technique and the $M4$ MOTIF compared to 0.2 and 0.467 respectively when using the $M13$ MOTIF. These results are the only ones that show a significant difference and highlight how much the choice of the MOTIF could impact the quality of the clustering. The traditional GMM based clustering also scored a fairly better ARI and $purity$, 0.513 and 0.677 respectively, and outperformed the MOTIFS technique in the clustering of this data set.

- The produced clusterings for $NADH-S1L$ and $NADH-S2L$ are all invalid, and for both MOTIFS, since the scored ARI is close to 0. The results were also similar to the ones produced with the traditional GMM. Indeed, these two datasets contain a significantly larger number of sequences when compared to the remaining experimental sets. This larger number of sequences could possibly infer a higher degree of overlapping among the clusters of these divergent datasets, and this could be leading to the failed GMM clustering. Moreover, the last computation step of the MOTIFS based technique uses the GMM to predict the optimal cuts. Therefore, it seems to inherit the deficiencies of the traditional GMM.

- The clustering results of the $COV2-Spike-UK$ data sets were perfectly identical for both $M4$ and $M13$ MOTIFS. When using the $Auto$ technique, these results matched the ones produced by the traditional GMM using MUSCLE or MAFFT for the alignment. Conversely, they slightly outperformed the ones achieved by the traditional GMM with EDLIB. Indeed, the MOTIFS approach with the $Auto$ technique produced a clustering with an ARI of 0.744 while the the GMM with EDLIB just obtained 0.691 that matches the results of the MOTIFS approach with the $Deltas$ technique. The clusterings with the best ARI also got the highest $purity$ level of 0.85.

- When clustering the $COV-Spike-4$ dataset, the $Deltas$ technique for the MOTIFS approach with either the $M4$ or $M13$ MOTIF, and the traditional GMM gave the best clusterings. The clustering that was produced using the AVG technique, grouped the COV and $COV2$ sequences into two pure clusters and produced a third cluster containing a singleton. Moreover, the advantage of the $Deltas$ technique that uses a non constant threshold, is highlighted by its superior results in the case of hybrid data sets.

- Finally, the $Deltas$ technique with the $M4$ MOTIF scored the best ARI of 0.654 and the highest $purity$ of 0.750 for this last hybrid $COV-Spike-5$ dataset, followed by the same approach with the $M13$ MOTIF and in the third place the traditional GMM with EDLIB. The latter only scored a slightly better ARI and $purity$ of 0.828 and 0.892 when the sequences were aligned with MUSCLE or MAFFT. Similarly to the clustering results of the previous dataset, the MOTIFS approach successfully

grouped the COV, $COV2$, and $MERS$ sequences into 3 clusters when using the AVG technique for both $M4$ and $M13$ MOTIFS.

These results present an additional proof of validity for the MOTIFS approach. They also show that our novel idea of establishing the graph links, based on a certain threshold, presents an effective solution for applying the MOTIFS approach in clustering biological sequences. Moreover, the results show that this method competes well with the traditional EM-GMM based approach, and might outperform it if a further enhancement is introduced in the selection of the threshold. Indeed, the correct choice of the threshold plays a crucial role in the MOTIFS approach and a wrong choice can drastically deteriorate the results as shown in the experiments. Conversely, a lightweight and automatic technique for an accurate choice of the threshold is necessary, because the presented algorithms in [Matar et al., 2019] and [Matar et al., 2021] fully automates the computation of the optimal parameters for the traditional GMM.

7.4.2/ DBSCAN

DBSCAN in another algorithm that enables the detection of randomly-shaped clusters. This feature makes it a potential concurrent to the state of the art algorithms that are used for clustering biological sequences. Several implementations of DBSCAN are available online, therefore, we adopted one of the freely available ones[4] for our experiments. Similarly to what was done with the threshold choices in the MOTIFS approach, a range of $Epsilons$ between 0.01 and 0.10, with a step equal to 0.01, was used in DBSCAN's experiments. Table 7.7 presents the parameters that gave the best ARI when clustering the different datasets with DBSCAN.

Data set	Epsilon	Nb. clusters	$1 < \|\|Cluster\|\| < 5$	Singletons	Purity	ARI
NADH-S1	0.07	4	0	0	1	1
NADH-S1L	0.04-0.05	5	0	0	1	0.957
NADH-S2	0.02	82	21	54	1	0.697
NADH-S2L	0.02	49	14	29	0.834	0.534
COV2-Vars	0.03	12	2	4	1	0.609
COV2-Spike-UK	0.04-0.05	3	0	1	0.850	0.691
COV-Spike-4	0.04-0.05	4	0	1	0.880	0.769
COV-Spike-5	0.04	5	0	1	0.892	0.791

Table 7.7: Clustering results using DBSCAN.

The computed ARI and $purity$ indexes in Table 7.7 show that DBSCAN produced good clustering results for all the datasets. Conversely from the previously obtained results by using the GMM-based approaches, DBSCAN successfully clustered the simulated $NADH$ datasets that present a very high level of divergence. The produced clusters for these

[4]that can be downloaded from https://github.com/james-yoo/DBSCAN

datasets were all perfectly pure except for the clustering of the $NADH-S2L$ dataset that presented a reduced $purity$ of 0.834 compared to the other results. An additional aspect of observation is the number of small clusters and the number of singletons (noise) in the results of DBSCAN. While there is no small clusters and singletons detected in the clustering results of $NADH-S1$ and $NADH-S1L$, a significant amount exists in the clustering results of $NADH-S2$ and $NADH-S2L$, that present a higher degree of gaps, insertions, and mutations. Indeed, this higher degree of changes in the sequences inferred the detection of micro clusters around the initial ones in addition to the noise-classified sequences. In the case of $NADH-S2$, and out of the 82 produced clusters, there was only 7 large clusters and 75 micro-clusters and noisy ones. A similar observation can be made to the best clustering result of $NADH-S2L$ where only 16 large clusters exist out of the 49 that were produced. The larger number of sequences in $NADH-S2L$ also generated a significantly higher number of clusters when compared to the expected four clusters.

In the case of the highly similar datasets $COV2-Vars$ and $COV2-Spike-UK$, the GMM-based approaches outperformed DBSCAN by scoring a slightly higher ARI. The best clustering results of $COV2-Vars$ scored an ARI of 0.718 for the traditional GMM, and 0.667 for the $M4$ MOTIFS-based GMM, compared to 0.609 for DBSCAN that produced a total of 6 small clusters or singletons out of its 12-clusters result. For this same data set, a perfect $purity$ is observed for the best results that were produced by the traditional GMM or DBSCAN, compared to 0.941 for the result of the MOTIFS approach with the $M4$ MOTIF. Concerning the $COV2-Spike-UK$ dataset, DBSCAN scored an ARI of 0.691 as the traditional GMM, but slightly lower than the $M4$ and the $M13$ MOTIF-based GMMs that scored a 0.744 ARI. The results with the highest ARI all scored an equal purity of 0.850, that implies a better resulting number of clusters for a better ARI. Indeed, the perfect number of clusters (3 for $COV2-Spike-UK$), that was produced by the traditional GMM, did not reflect a perfect grouping. It was rather composed of a cluster, consisting of the merger of two clusters from the perfect clustering, and two other clusters split from another cluster from the perfect clustering. Conversely, the MOTIFS-based GMM produced 2 clusters: a perfect one and another one merging two clusters from the prefect clustering, and containing the spike proteins of 2 close lineages.

Conversely, in the case of the hybrid datasets $COV-Spike-4$ and $COV-Spike-5$, DBSCAN outperformed the GMM-based approaches (using EDLIB) in both ARI and $purity$ metrics. For $COV-Spike-4$, the clustering produced by DBSCAN scored an ARI and a $purity$ equal to 0.769 and 0.880 respectively. The GMM-based approaches scored slightly lower ARI and $purity$ indexes, 0.733 and 0.840. Concerning the $COV-Spike-5$ dataset, the clustering with DBSCAN gave the highest ARI and $purity$ indexes of 0.791 and 0.892 respectively. It was followed by the MOTIFS approach with the $M4$ and the $M13$ MOTIFS. The traditional GMM scored a significantly worse ARI of 0.355 and a slightly worse $purity$

of 0.607. Therefore, it can be concluded that DBSCAN outperformed the GMM for the very highly divergent datasets and the hybrid ones, while the GMM preserved the upper hand for the highly similar ones. But similarly to the MOTIFS-based approach that requires a delicate choice of threshold, DBSCAN also requires a delicate choice of $Epsilon$ to produce a good quality clustering.

7.4.3/ HDBSCAN

With a variable and dynamically computed $Epsilon$, HDBSCAN suppresses the need of a user input $Epsilon$ when compared to its predecessor DBSCAN. HDBSCAN also has the ability of detecting random-shaped clusters in addition to detecting clusters with variable densities. These properties are advantageous for clustering biological sequences where the degree of mutations is unpredictable. Similarly to DBSCAN, many implementations of HDBSCAN exist. We also adopted a freely available implementation[5] for our next experiments.

Since HDBSCAN requires a user-input of the minimum number of elements in a cluster, i.e., the minimum cluster size below which the elements will be considered as noise, this parameter is set in our experiments as follows:

- For the 4 simulated $NADH$ datasets, the minimum cluster size is set to 150, then 100, and finally 50. If the scored ARI remains at its maximum upon reaching 50, then we keep reducing the minimum cluster size by 10 at each step until a lower ARI is scored.
- For the $COV2-Vars$ dataset, the minimum cluster size is initially set to 21, the size of the smaller cluster in the reference clustering. It is then decreased by 1 at each step until a lower ARI is recorded.
- The minimum cluster size is set to 3 for the remaining datasets since this is the size of the smaller cluster (the one grouping the $COV2$ spike protein sequences of the B.1.1 lineage).

It should be noticed that the large step of 50 in the choice of the minimum cluster size, that is used in the first 3 experiments on the $NADH$ data sets, does not cause a drastic impact on the results; for example, the choice of 100 instead of 150 in the case of $NADH-S1L$ only produces 1 additional cluster that is reflected in a slight decrease of 0.074 in the ARI, and without affecting the purity. The clusterings that scored the highest ARI are presented in Table 7.8.

[5] downloadable at https://github.com/rohanmohapatra/hdbscan-cpp

Data set	Min. clust. size	Nb. clusters	$1 < \|\|Cluster\|\| < 5$	Singletons	Purity	ARI
NADH-S1	100	4	0	0	1	1
NADH-S1L	150	77	0	73	1	0.953
NADH-S2	150	286	0	283	1	0.766
NADH-S2L	30-150	47	0	40	1	0.734
COV2-Vars	20	10	0	6	1	0.712
COV2-Spike-UK	3	6	3	2	1	0.725
COV-Spike-4	3	7	3	2	1	0.787
COV-Spike-5	3	7	4	1	1	0.835

Table 7.8: Clustering results using HDBSCAN.

The clusterings generated by HDBSCAN are largely similar to the ones returned by DBSCAN. HDBSCAN outperforms the GMM-based approaches when applied on the highly divergent datasets and on the hybrid ones, while it is less efficient in the case of the highly similar datasets. Moreover, HDBSCAN slightly outperforms its predecessor DBSCAN in all the results except the one of $NADH-S1L$ for the ARI score. In this case, where DBSCAN scored an ARI of 0.957 vs. 0.953 for HDBSCAN, this index penalized HDBSCAN for its high number of identified noise points, 73 besides the 4 correct clusters. Conversely, DBSCAN was penalized for splitting a cluster in two and producing 5 large clusters, but without any noise. Moreover, HDBSCAN produced significantly more singletons (noise) in its experiments, when compared to DBSCAN. The highest number of singletons, 283, was obtained in the clustering result of $NADH-S2$.

An additional and remarkable property in the results of HDBSCAN is that all the clusterings that gave the best ARI also obtained a perfect $purity$. This observation was not encountered with the previous techniques. In addition, HDBSCAN is so far the only algorithm in the experiments that succeeded in correctly grouping the $COV2$ spike protein, of the B.1.1 lineage, into a separate cluster.

7.5/ INTRODUCING THE CHAINS CLUSTERING TECHNIQUE

After evaluating the state of the art clustering techniques, in this section, we introduce the CHAINS technique, a novel, easy to implement, and computation-efficient technique for clustering biological sequences.

CHAINS is a single linkage technique that performs the following two steps in its basic version:

1. Progressively visit each sequence and link it to its closest sequence (the one that has the highest pairwise similarity with it).

2. After visiting all the sequences, group the linked sequences, that became members of a same formed CHAIN, ending with a two-sequences loop, into a separate cluster.

Figure 7.2 illustrates the above steps for producing a clustering.

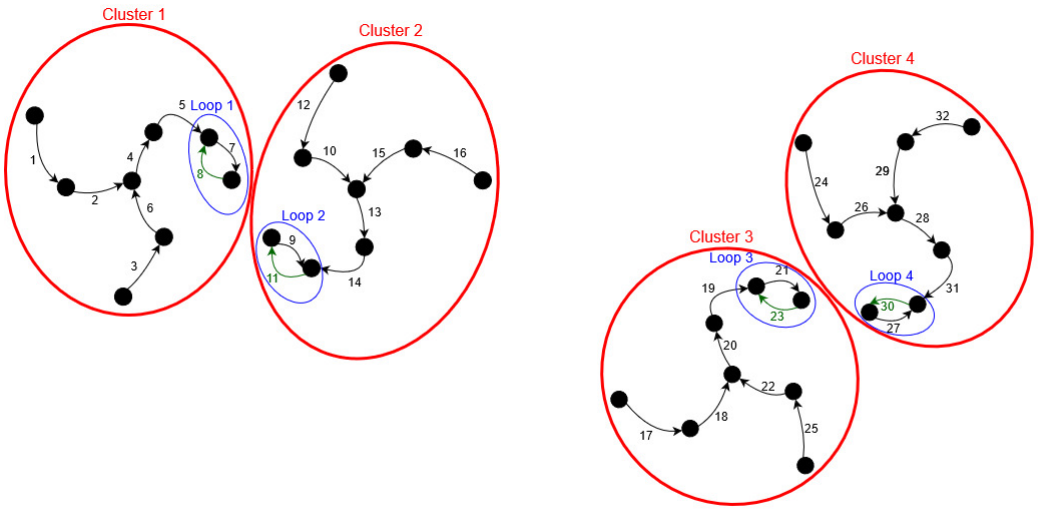

Figure 7.2: CHAINS with ending loop of 2 elements.

The first and simplest version, of our CHAINS technique, tends to naturally produce clusters that can be very close (see clusters 1 and 2, and clusters 3 and 4 in Figure 7.2). These close clusters should be merged in order to produce a better clustering, and avoid producing small sub-clusters of a same cluster. Therefore, we propose an enhanced version, that solves the previously highlighted potential anomaly. It consists of the following steps:

1. Input a user-defined minimum size for the ending loop mls, that should be less or equal to the number of input sequences.
2. For each sequence, establish a list of the neighboring sequences, of size $mls - 1$, and sort the list by descending similarity with the source sequence.
3. Progressively visit each sequence and link it to its closest neighbor (the one at index 0 in the neighbors list)
4. If a loop is formed following the established link in the previous step then:
    1. Count the number of sequences in the formed loop
    2. If this number is less than mls then delete the last established link and move it to the next closer neighbor (at the index $i + 1$ where i is the index of the previously linked neighbor in the list).
    3. Repeat the previous two steps if a new loop is formed.
5. When all the sequences were visited, group the linked sequences, that became members of the same CHAIN, that ends with a loop consisting of minimum mls-sequences, into a separate cluster.

Figure 7.3 illustrates the enhanced version of the CHAINS technique, with an mls equal to 4. The same cloud of points, that was used in the Figure 7.2, is preserved in order to illustrate the behavior change with the second version.

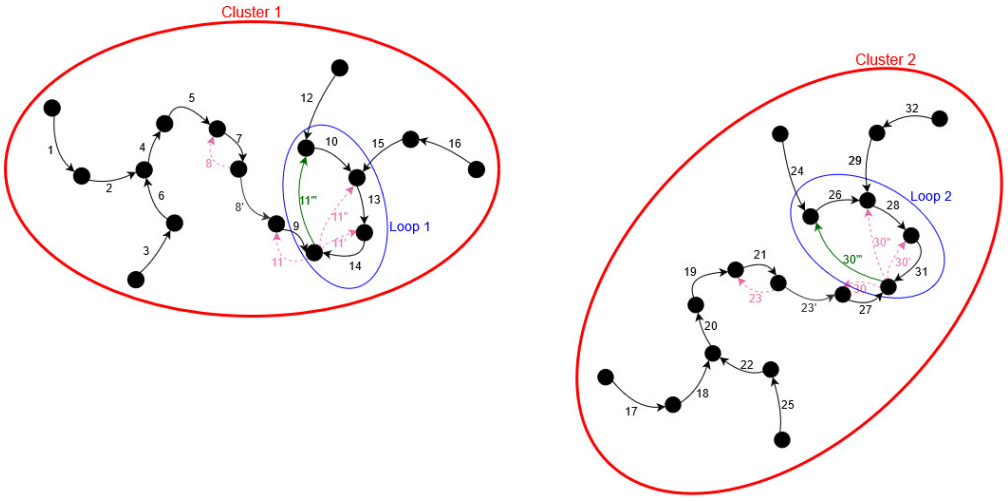

Figure 7.3: CHAINS with ending loop of 4 elements.

Visually, the two produced clusters, illustrated in Figure 7.3, are better formed and separated than the ones produced by the initial version of our algorithm and illustrated in Figure 7.2. The numbers beside the arrows in Figure 7.3 show the linkage order that was done using the enhanced CHAINS algorithm with $mls = 4$. The pink arrows illustrate the established, then deleted links, while the green arrows illustrate the last established links that led to the formation of an ending loop that fulfills the minimum required loop size. Finally, a number of clusters, matching the number of formed loops, are identified and produced. This novel chains technique allows the identification of random sized clusters, similarly to DBSCAN and HDBSCAN.

In order to compare the CHAINS technique to the previously tested ones, it was applied on the same datasets as in the previous experiments. The choice of the minimum loop size parameter was initially set to 2 and then incremented by 1 until obtaining three consecutive clusterings with decreasing ARI indexes or until the size of the smallest cluster in the reference clustering is reached. Table 7.9 presents the clusterings that were produced by the CHAINS technique and that scored the best ARI.

After analysing the minimum loop sizes that produced the best clusterings, we suspect a potential correlation between the best loop size and the divergence degree of the target dataset can be found. In fact, for the highly similar datasets, the optimal loop sizes were small, between 2 and 5. Conversely, it was big, between 14 and 22, for the simulated highly divergent datasets.

Data set	Min. loop size	Nb. clusters	Purity	ARI
NADH-S1	14-15	4	1	1
NADH-S1L	19-20	4	1	1
NADH-S2	18	6	1	0.893
NADH-S2L	22	7	1	0.818
COV2-Vars	4-5	7	0.991	0.596
COV2-Spike-UK	2-3	3	1	1
COV-Spike-4	3	3	0.880	0.809
COV-Spike-5	3	4	0.892	0.828

Table 7.9: Clustering results using the CHAINS approach.

According to the computed $purities$ and ARI indexes, the CHAINS technique outperformed its competitors in clustering all the considered datasets except for the $COV2-Vars$ and $COV-Spike-5$ datasets. In addition, CHAINS is the only tested technique that perfectly clustered the $NADH-S1L$ and the $COV2-Spike-UK$ datasets. Only the best clustering of HDBSCAN scored a slightly higher ARI of 0.835, compared to 0.828 for CHAINS, in the case of $COV-Spike-5$. For the $COV2-Vars$ dataset, despite scoring the lowest ARI among the other techniques, the best clustering result produced using CHAINS has a high $purity$ 0.991 and produced 7 clusters instead of the expected 3 clusters, therefore this result shows that the CHAINS technique mainly divided the perfect clusters into multiple sub-clusters in this case.

7.6/ COMPARISON WITH SOME STATE OF THE ART CLUSTERING TOOLS

In this section, we present a comparative study between the previously assessed techniques (GMM, DBSCAN, HDBSCAN and CHAINS) with some of the state of the art biological clustering tools such as CD-HIT, UCLUST and AncestralClust.

7.6.1/ QUALITATIVE COMPARISON

CD-HIT, UCLUST and AncestralClust are the three main competitors to SpCLUST in the field of clustering divergent sequences. To compare them to the previously assessed techniques, they were applied to the same eight datasets described in Section 7.2. CD-HIT and UCLUST require a sensitive choice of a similarity or identity threshold, while AncestralClust takes an user-defined number of initial sequences for creating the initial neighbor joining trees. If this parameter is not user defined for AncestralClust, it takes the default value, 10. Moreover, this parameter influences the number of resulting clusters. Therefore, the parameters of CD-HIT and UCLUST were chosen for each dataset by trial and error with a threshold precision of 0.001. On the other hand, AncestralClust was

tested using the default value (10), then with an input of the actual correct number of clusters in the reference clustering.

Although AncestralClust is intended to cluster nucleotide sequences, the experiments showed that it is not designed to handle non-ATCG bases. Therefore, as a workaround, the rare occurrences of K, Y, R, S and M bases, existing in the sequences of the COV2-Vars dataset, were replaced with N[6] in order to perform the clustering with AncestralClust. In addition, conversely from the other state of the art tools, the sequences in the datasets have to be formatted as a single line per sequence in order to be accepted by AncestralClust. Table 7.10 presents the clusterings produced by the three considered tools and that scores the highest ARI indexes.

Data set	Clustering tool	Threshold/Init. c. nb.	Nb. clusters	Singletons	Purity	ARI
NADH-S1	CD-HIT	0.800	148	6	-	-
NADH-S1L	CD-HIT	0.800	234	8	-	-
NADH-S2	CD-HIT	0.800	760	543	-	-
NADH-S2L	CD-HIT	0.800	1551	1285	-	-
COV2-Vars	CD-HIT	0.995	7	3	0.815	0.414
COV2-Spike-UK	CD-HIT	0.995	3	1	0.850	0.691
COV-Spike-4	CD-HIT	0.995	5	2	0.880	0.731
COV-Spike-5	CD-HIT	0.995	7	3	0.892	0.739
NADH-S1	UCLUST	0.650	4	0	1	1
NADH-S1L	UCLUST	0.700	4	0	1	1
NADH-S2	UCLUST	0.360	66	2	-	-
NADH-S2L	UCLUST	0.370	84	1	-	-
COV2-Vars	UCLUST	0.999	5	2	1	0.979
COV2-Spike-UK	UCLUST	0.997	3	1	0.900	0.801
COV-Spike-4	UCLUST	0.994	3	0	0.880	0.809
COV-Spike-5	UCLUST	0.994	5	1	0.892	0.812
NADH-S1	AncestralClust	Default	6	0	1	0.905
NADH-S1	AncestralClust	4	4	0	1	1
NADH-S1L	AncestralClust	Default	7	0	1	0.945
NADH-S1L	AncestralClust	4	3	0	0.750	0.627
NADH-S2	AncestralClust	Default	9	0	0.577	0.158
NADH-S2	AncestralClust	4	8	0	0.579	0.177
NADH-S2L	AncestralClust	Default	7	0	0.481	0.091
NADH-S2L	AncestralClust	4	5	0	0.413	0.074
COV2-Vars	AncestralClust	Default	10	2	1	0.599
COV2-Vars	AncestralClust	3	3	0	1	1

Table 7.10: Clustering results using CD-HIT, UCLUST, and AncestralClust.

The followings are the main observations obtained from the results presented in Table 7.10:

- Although its minimum supported similarity threshold was selected, CD-HIT failed to produce any significant clustering results for the highly divergent $NADH$ datasets. The sequences of these sets were in majority clustered in very small clusters or as singletons.
- Since UCLUST supports lower identity threshold values, it successfully clustered

[6]N is treated by AncestralClust as an unidentified base.

the divergent sets $NADH-S1$ and $NADH-S1L$, but it failed to cluster $NADH-S2$ and $NADH-S2L$ that present a higher frequency and sizes of insertions and gaps. The used thresholds, for clustering the $NADH-S2$ and $NADH-S2L$ sets, are the ones that produced the smaller number of clusters. Below these thresholds, UCLUST unexpectedly produced a higher number of clusters.

- UCLUST outperformed CD-HIT in all the experiments. It outperformed AncestralClust as well except for the experiments on the following datasets:
  - $NADH-S1$ where both UCLUST and AncestralClust produced a perfect clustering.
  - $COV2-Vars$ where AncestralClust produced a perfect clustering compared to a closely perfect one that was produced by UCLUST, and that contained only 2 wrongly clustered sequences as singletons.
  - $NADH-S2$ and $NADH-S2L$ where UCLUST produced a very high number of clusters compared to the expected one. AncestralClust also produced a clustering that scored a maximum ARI as low as 0.177 for $NADH-S2$, and an invalid clustering scoring a maximum ARI close to zero for $NADH-S2L$.
- AncestralClust produced a number of clusters that is equal or closely equal to the expected one, when this number was manually provided as its initial number of clusters. Conversely, the purity of its clustering declined in some cases following this choice. In these cases, the scored ARI was also lower than the one scored by using its default setting.

A comparison between the results in Table 7.10, and the best ones produced by the previously assessed techniques shows the following:

- In the case of $NADH-S1$, UCLUST and AncestralClust compete well with the DBSCAN, HDBSCAN, and CHAINS techniques, by producing a perfect clustering.
- Only UCLUST produces a perfect clustering, similarly to the CHAINS technique, in clustering $NADH-S1L$.
- A good clustering of the $NADH-S2$ and $NADH-S2L$ is only attainable by using the DBSCAN, HDBSCAN, and CHAINS techniques. The best clustering is obtained by using the CHAINS'.
- The recent tool AncestralClust was the only tool that could perfectly cluster the $COV2-Vars$ dataset. Moreover, UCLUST was not so far from producing such a perfect clustering. The use of the traditional EM-GMM in SpCLUST also produced a fairly good clustering that scored a 0.718 ARI for this dataset, when aligning the sequences with EDLIB.

- Although UCLUST and CD-HIT produced a fairly good clustering for $COV2-Spike-UK$, all the other techniques were able to produce similar or even better results. Furthermore, only the CHAINS technique produced a perfect clustering for this dataset.
- Fairly good clustering results for $COV-Spike-4$ were produced by CD-HIT, UCLUST, in addition to any one of the previously assessed techniques. For this data set, both UCLUST and the CHAINS technique produced the best clustering that scored the highest ARI.
- Finally, the clustering experiments on $COV-Spike-5$ also show good results from all the parties. The highest quality results were recorded by the HDBSCAN technique, closely followed by the CHAINS'.

While multiple techniques or tools could compete in clustering a single dataset, this comparative study further supports our claim that none of these techniques can claim its superiority in all possible cases. Indeed, UCLUST and AncestralClust along with DBSCAN, HDBSCAN, and CHAINS are all able to produce valid clusterings in the case of moderately divergent datasets, regardless of the number of input sequences. Conversely, UCLUST and AncestralClust fail in clustering highly divergent datasets. Yet, all the assessed tools and techniques were able to produce valid clusterings for the highly similar datasets. The CHAINS technique proved to be superior than the other considered tools in most of the studied cases.

Beside the quality of the produced clustering, the clustering speed is another important aspect when comparing clustering tools. In the next section, the clustering speed of all these tools is compared.

7.6.2/ CLUSTERING SPEED COMPARISON

The experiments show that, for all the datasets that were used, the clustering time does not exceed a few seconds when the traditional tools CD-HIT and UCLUST are used. Similarly, when using the rest of the considered clustering techniques, the total processing time, for all the phases that are involved in the clustering pipeline, is less than 1 second for the $COV2-Spike-UK$, $COV-Spike-4$, and the $COV-Spike5$ datasets. However, for the remaining datasets, AncestralClust took the least amount of time to cluster them. The traditional GMM and the MOTIFS approaches both recorded highly similar times. The best recorded one among both techniques is listed under GMM. Table 7.11 displays the observed clustering times and Figure 7.4 illustrates them.

The results show that the traditional tools, namely CD-HIT and UCLUST, remain faster than the other ones. The performance comparison over the large data sets, shows the following:

Clustering tech.	COV2-Vars	NADH-S1	NADH-S2	NADH-S1L	NADH-S2L
GMM	54	115	134	1040	1072
DBSCAN	52	118	125	936	981
HDBSCAN	56	109	199	1038	1171
CHAINS	52	22	38	87	158
AncestralClust	378	2	4	3	10
UCLUST	1	1	1	1	1

Table 7.11: Processing time in seconds (using an Intel i7-6700 CPU).

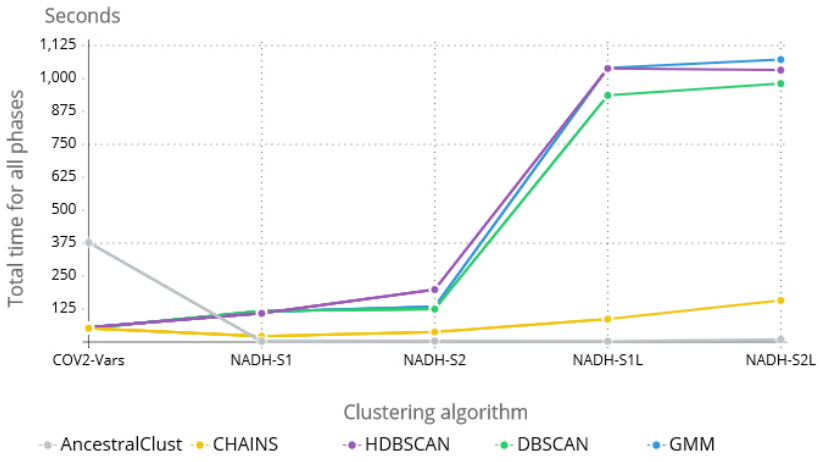

Figure 7.4: Processing time chart.

- AncestralClust requires significantly more time to cluster the datasets that contain very large sequences, such as the $COV2{-}Vars$ that contains full genomes of $SARS{-}COV2$. Conversely, AncestralClust outperforms the assessed clustering techniques, speed-wise, if the clustered sequences are smaller, and regardless of their number. This may be due to the alignment algorithm that is used in AncestralClust.
- The GMM-based techniques along with DBSCAN and HDBSCAN require similar execution times because they mainly share the same initial phases that are significantly more time consuming that the clustering phase itself.
- The CHAINS technique outperforms the other ones with a speed up reaching up to 11.95× in the case of clustering the $NADH - S1L$ dataset. The DBSCAN and the CHAINS techniques outperformed AncestralClust in clustering the $COV2 - Vars$ dataset with a speed up of 7.26×.

Moreover, a detailed profiling for the implementation of the CHAINS technique shows that the processing time of the final clustering phase of its pipeline, using the CHAINS algorithm, did not exceed 1 second for all the datasets. In this case, the recorded total processing time is mainly consumed by the alignment and the similarity matrix calculation phase. Conversely, for the other options, the required time for this same phase changes depending on the input parameters, such as the $Epsilon$ for DBSCAN and the minimum

cluster size for HDBSCAN, but did not exceed the ceil of 10 seconds. This proves that CHAINS, our newly introduced technique, outperforms the other ones in this phase.

7.7/ CONCLUSION

In this chapter, a qualitative study of several clustering techniques and tools for biological sequences was presented. Novel algorithms were proposed to produce faster and higher quality clusterings for potentially divergent sequences.

First of all, a new library that enables performing a fast and memory efficient pairwise sequences alignment, was suggested for boosting the overall clustering speed of GMM based tools. This new library also enables them to handle very large sequences, such as full genomes. Performance-wise, the experiments on large data sets show that this library allowed a speed up reaching 1226.38× in the computation of the similarity matrix, when compared to MUSCLE and MAFTT.

Second, the MOTIFS-based clustering approach was adapted to the biological sequences clustering context and novel threshold selection techniques were proposed to generate the graph connecting close sequences. The experiments demonstrated that the MOTIFS approach is a serious competitor to the traditional EM-GMM.

Third, a new clustering technique called CHAINS was proposed and it allowed a speed up of up to 11.95×, when compared to the other clustering techniques and it scales well when clustering large datasets. Its main drawback is that it cannot detect singleton clusters. The experiments showed that HDBSCAN and CHAINS were the only techniques able to successfully cluster datasets containing highly mutated sequences.

Finally, a comparative study between various clustering techniques and tools was conducted. It was discovered that many clustering techniques or tools are valid for each one of the different assessed levels of insertions and mutations. Yet, few are the ones that produce the best clustering quality in each case. Our novel CHAINS technique proved its efficiency in all the studied cases and shared the top ranking with a couple of different techniques in particular cases. Conversely, the traditional tools remain the fastest and the best suited options for clustering highly and moderately similar sequences, especially when the time factor overweight any small quality improvements.

The following part of this thesis concludes our work by further elaborating our findings. It also presents our future perspectives.

IV

Conclusion & Perspectives

8

CONCLUSION & PERSPECTIVES

8.1/ CONCLUSION

In this thesis, clustering approaches as well as processing speed optimization techniques have been proposed and assessed in order to ensure a fast and reliable clustering for biological sequences, having an unknown degree of similarity. This dissertation is composed of two parts: the first part covers a brief review of the state of the art clustering methods and tools, and their application to biological sequences, along with the clustering validation metrics, whereas the second part presents the contributions that have been made in this thesis.

The first part began by presenting some existing clustering techniques that are used in a wide diversity of applications in the literature. Some of the technical aspects of these techniques were considered such as their required input parameters, the shapes of the clusters that they can possibly detect, etc. These aspects serve as the basic criteria for the choice of their suitable application fields.

In the next chapter, several existing tools, designed for clustering biological sequences, were presented. The required input parameters and the targets of these tools were discussed. Moreover, their main drawbacks that need to be addressed, were also identified.

Given that the main goal of this thesis is the advancement of the biological sequences clustering, assessing the resulting clusterings in our experiments is crucial. Therefore, several clustering validation techniques and indexes were presented at the end of this first part. The advantages and drawbacks of the most common internal and external validation indexes were described.

The second part of this dissertation presented the contributions. The first research work focused on the optimization of a recently proposed pipeline that uses GMM for clustering potentially divergent sequences. An efficient and fast clustering package for divergent biological sequences, called SpCLUST, was proposed. This hybrid package uses different modules in its pipeline, which involves a third-party tool for aligning the sequences,

a C++ module for computing an affinity matrix, and a Python module for implementing the unsupervised learning method, namely the EM-GMM. The C++ module uses MPI to compute in parallel the affinity matrix. The performance tests showed that this module offers a 167.5X speed-up over the original Python module. Subsequently, the new hybrid package delivered a speed-up ranging from 37.9X to 44.6X when compared to the original one. Moreover, by including two additional substitution matrices for the distance computation, PAM250 and BLOSUM62, the scope of usage for this new package was also extended to handle protein sequences.

SpCLUST was intensively tested by using 26 different datasets, including real sequences and simulated sequences that represented various degrees of similarity or divergence. In both categories of the sequences, genomic and protein sets were included to cover the two scopes. The results of our experiments showed that SpCLUST successfully clustered these sets without any a priori knowledge of the number of clusters and without requiring a similarity threshold. Conversely, a comparative study between SpCLUST and some widely used clustering tools demonstrated that these tools require a delicate choice of the similarity or identity threshold in order to produce a valid clustering.

Despite proving that the use of the GMM along with the BIC provides a good clustering tool for potentially divergent sequences that does not require any user intervention, and despite the huge performance improvement introduced in SpCLUST when compared to the previous implementation of the algorithm, the complexity of this model remains by far higher than the complexity of the traditional greedy or hierarchical ones. Therefore, the proposed parallel tool remains much slower than the traditional High Performance Computing tools based on hierarchical algorithms.

The second research that was done in this thesis focused on introducing further enhancements to the GMM-based approach that was adopted in SpCLUST, in addition to presenting a comparative study between SpCLUST and some sequence alignment tools. The efficiency of using the GMM in clustering hybrid sequences, in addition to sets of sequences that were subject to horizontal gene transfers, is also tackled. The introduced enhancements included a technical part for the speed optimization, by porting the remaining Python module to C++. Alongside with this technical part, three GMM-based algorithms were proposed and implemented, in addition to three additional types of affinity matrices. These algorithms and matrices introduce new options for the user in the new package called SpCLUST-V2.

The introduced GMM-based algorithms focus on an iterative process that uses the GMM and changes its initial seed at each iteration. Because the choice of this seed can lead to a different convergence state of the GMM, the user can choose between using a single and random seed, or iterating with a defined number of different seeds. In the iterative approach, the user can finally choose either the clustering scoring the best BIC or the

clustering that scores the highest number of occurrences in the iterations. A performance comparison between SpCLUST and SpCLUST-V2 on a large dataset showed that the new version offered a speed-up reaching 42X in the clustering phase, and based on the chosen algorithm.

Our conducted experiments involved 7 different datasets, in which four sets are hybrid and present a simulation of horizontal gene transfer. These experiments were used to assess the newly introduced algorithms, along with the newly embedded types of affinity matrices. The results of the experiments proved that our new package successfully clusters the selected datasets, and is more capable than the state of the art tools in clustering the ones containing different types of genomes. Contrary to phylogenetic trees, SpCLUST-V2 is also capable of successfully handling the cases of horizontal gene transfer. In general, the best clustering results are mainly obtained when using the best BIC algorithm with at least one of the embedded affinity matrices.

Finally, the comparative study between the selected alignment tools, concluded that the alignment produced by these tools does not have any significant effect on the quality of the clustering, in the case where the involved sequences are small and highly similar. Conversely, MUSCLE and MAFFT are the best suited for aligning divergent sequences, while MAFFT keeps the upper hand in handling large ones where MUSCLE fails or requires huge memory resources.

In the third research, a comparative study between various clustering techniques and tools was conducted and a further improvement is introduced to the sequences alignment stage. Starting with the alignment, the adoption and implementation of a new library enabled a fast and memory efficient alignment that tremendously boosts the overall speed of the GMM-based tools, without any significant degradation of the clustering quality. The adoption of this library also enabled the handling of very large sequences, where the previously used alignment tools either failed to align large sequences or took a lot of time. In addition, we highlight that the alignment and the pairwise similarity computation phases are essential in our clustering pipeline, regardless of the chosen clustering technique.

The comparative study included some carefully selected clustering techniques, namely the DBSCAN, HDBSCAN, and MOTIFS, in addition to our novel one, the CHAINS. The selection was made based on the shape of the clusters these techniques can detect, in order to judge if they are suitable for clustering biological sequences in terms of clustering quality and speed:

- Quality-wise, it was discovered that the DBSCAN, HDBSCAN, and AncestralClust cluster well sequences that undergone a moderate level of insertions and mutations, but UCLUST and the CHAINS technique are more suited for such datasets. Conversely, CHAINS is the recommended technique to be used where a high level

of insertions and mutations exist, although the DBSCAN and HDBSCAN are also applicable in this case. All the GMM-based techniques, in addition to UCLUST and AncestralClust, are capable of handling highly similar nucleotide sequences, but the use of AncestralClust is advised for clustering such sequences. Except for AncestralClust that only handles nucleotide sequences, our experiments showed that all the tested tools and techniques are useful for clustering the highly similar protein sequences, yet the CHAINS technique is more recommended. For the datasets consisting of sequences from different species, all the tested tools and algorithms performed well in our experiments. However, the HDBSCAN and CHAINS techniques, in addition to UCLUST, proved to be the best suited to this variable density case: they detected better the different lineages in the same species in addition to correctly splitting sequences from different species.

- Speed-wise, the traditional tools CD-HIT and UCLUST remain the fastest, although the newly introduced CHAINS technique outperforms the rest of the considered techniques. Despite the performance improvement that was achieved in the alignment stage, the GMM based clustering approaches remain more complex and compute intensive than the traditional tools. Therefore, the traditional tools are the best option for clustering highly and moderately similar sequences, with a well known similarity threshold, especially when the time factor overweight any small quality improvements.

8.2/ PERSPECTIVES

This thesis tackled an important and ongoing research challenge: the analysis of biological sequences. Therefore, it proposed novel solutions for clustering the biological sequences in general and the potentially divergent ones in particular. The results obtained in this thesis led us to the discovery of multiple perspectives that we would like to pursue and develop in the near future. Moreover, multiple steps of the clustering pipeline could be further optimized, starting form the similarity computation techniques, the eigensolvers, the implementation of the GMM, the threshold computation algorithms for the MOTIFS approach, and the automation of the selection of the optimal parameters for the clustering techniques. A visual representation of the clustering result should also be added to the package.

The pairwise similarity computation among the sequences is an essential part for their analysis and clustering. This part is still a serious obstacle for improving the scalability of our presented approaches because it requires the alignment of the sequences. Indeed, only few algorithms are proposed for an alignment-free sequence comparison, and a single one only is claimed producing biologically relevant scores [Girgis et al., 2021]. There-

fore, investigating the accuracy of these proposed algorithms, or defining novel ones for calculating the pairwise similarities, without the need of aligned sequences, can further enhance the speed and the scalability of the clustering.

Moreover, the eigensolvers are essential for computing the data embedding (eigenmap) for the spectral clustering techniques. Therefore, an additional possible extension to our work could be the assessment of the speed and accuracy of the state of the art eigensolvers [Demmel, 1991, Sanderson et al., 2016, Guennebaud et al., 2010]. Beside the existing parallel schemes for computing the general eigenvalues problem [Auslander et al., 1992], proposing an enhanced parallel eigensolver based on Jacobi's algorithm [Sameh, 1971] would also improve the speed of the overall process of spectral clustering and increase the scalability of our proposed tool. These propositions are also valid for the implementation of the GMM that was proved being an efficient clustering technique in the spectral approaches.

Furthermore, it is essential to find a more efficient and intervention-free algorithm for computing and applying the adequate thresholds in the MOTIFS approach. This recent GMM-based approach presents a promising solution for clustering large graphs. In this same track of eliminating the user intervention, automated algorithms are required for finding the appropriate parameters for the HDBSCAN and CHAINS methods. Moreover, a further enhancement to CHAINS should enable the detection of correct singleton clusters.

Finally, beyond the clustering of biological sequences, applying a certain distance threshold for eliminating some links, prior to clustering other types of graphs, could possibly enhance the results of the MOTIFS approach. Moreover, the new CHAINS clustering approach should be applied in other fields where it might prove to be very efficient.

PUBLICATIONS

JOURNAL PAPERS

- Johny Matar, Hicham El Khoury, Jean-Claude Charr, Christophe Guyeux, Stéphane Chretien. *"SpCLUST: Towards a fast and reliable clustering for potentially divergent biological sequences"*. In **Computers in Biology and Medicine**, 114, 103439 (2019).

SUBMITTED PAPERS

- Johny Matar, Hicham El Khoury, Jean-Claude Charr, Christophe Guyeux, Stéphane Chretien. *"Biological sequence clustering: novel approaches and a comparative study"*. In **Journal of Computational Science**. Submitted in August 2021.
- Johny Matar, Hicham El Khoury, Jean-Claude Charr, Christophe Guyeux, Stéphane Chretien. *"Optimized spectral clustering methods for potentially divergent biological sequences"*. In **IEEE/ACM Transactions on Computational Biology and Bioinformatics**. Submitted in November 2021.

IN PROCEEDINGS

- Johny Matar, Hicham El Khoury, Jean-Claude Charr, Christophe Guyeux. *"Impact of eigensolvers on spectral clustering"*. In **IEEE MENACOMM'21**

BIBLIOGRAPHY

[GMM,] **8.18.1. sklearn.mixture.gmm**. https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/generated/sklearn.mixture.GMM.htmll. Accessed: 2019-12-27.

[arm,] **Armadillo: C++ library for linear algebra & scientific computing**. http://arma.sourceforge.net/docs.html#gmm_full. Accessed: 2020-02-18.

[cal,] **Calinski-harabasz index and boostrap evaluation with clustering methods**. https://ethen8181.github.io/machine-learning/clustering_old/clustering/clustering.html. Accessed: 2019-10-08.

[clu,] **Cluster analysis - wikipedia**. https://en.wikipedia.org/wiki/Cluster_analysis. Accessed: 2021-02-27.

[cog,] **cogent.pypi**. https://pypi.org/project/cogent/. Accessed: 2018-10-11.

[pep,] **Github - paperrune/gmm: C++ implementation of gaussian mixture model**. https://github.com/paperrune/GMM. Accessed: 2019-12-02.

[Inf,] **Influenza**. https://en.wikipedia.org/wiki/Influenza. Accessed: 2019-02-12.

[kme,] **kmeans-clustering-cpp/kmeans.cpp**. https://github.com/aditya1601/kmeans-clustering-cpp/blob/master/kmeans.cpp. Accessed: 2019-06-07.

[lev,] **Levenshtein distance**. https://en.wikipedia.org/wiki/Levenshtein_distance. Accessed: 2019-03-07.

[MUS,] **Muscle user guide**. http://www.drive5.com/muscle/muscle.html. Accessed: 2018-09-27.

[EDN,] **Rosalind — glossary — dnafull**. http://rosalind.info/glossary/dnafull/. Accessed: 2018-09-27.

[Spe,] **sklearn.manifold.spectral_embedding**. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.spectral_embedding.html. Accessed: 2019-12-27.

[Gau, a] **sklearn.mixture.gaussianmixture**. http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html. Accessed: 2018-10-10.

[Gau, b] **sklearn.mixture.gaussianmixture**. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html. Accessed: 2019-12-27.

[Sub,] **Substitution matrix**. https://en.wikipedia.org/wiki/Substitution_matrix. Accessed: 2018-09-27.

[Afgan et al., 2018] Afgan, E., Baker, D., Batut, B., Van Den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., Guerler, A., Hillman-Jackson, J., Hiltemann, S., Jalili, V., Rasche, H., Soranzo, N., Goecks, J., Taylor, J., Nekrutenko, A., et Blankenberg, D. (2018). **The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update**. *Nucleic Acids Research*, 46(W1):W537–W544.

[Ahola et al., 2006] Ahola, V., Aittokallio, T., Vihinen, M., et Uusipaikka, E. (2006). **A statistical score for assessing the quality of multiple sequence alignments**. *BMC bioinformatics*, 7(1):484.

[Auslander et al., 1992] Auslander, L., et Tsao, A. (1992). **On parallelizable eigensolvers**. *Adv. Appl. Math*, 13:253–261.

[Benson et al., 2016] Benson, A. R., Gleich, D. F., et Leskovec, J. (2016). **Higher-order organization of complex networks**. *Science*, 353(6295):163–166.

[Biernacki et al., 2016] Biernacki, C., Castellan, G., Chretien, S., Guedj, B., et Vandewalle, V. (2016). **Pitfalls in mixtures from the clustering angle**. In *Working Group on Model-Based Clustering Summer Session*.

[Biernacki et al., 2003] Biernacki, C., et Chrétien, S. (2003). **Degeneracy in the maximum likelihood estimation of univariate gaussian mixtures with em**. *Statistics & probability letters*, 61(4):373–382.

[Bocklet et al., 2008] Bocklet, T., Maier, A., Bauer, J. G., Burkhardt, F., et Noth, E. (2008). **Age and gender recognition for telephone applications based on gmm supervectors and support vector machines**. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1605–1608. IEEE.

[Bruneau et al., 2018] Bruneau, M., Mottet, T., Moulin, S., Kerbiriou, M., Chouly, F., Chretien, S., et Guyeux, C. (2018). **A clustering package for nucleotide sequences using laplacian eigenmaps and gaussian mixture model**. *Computers in Biology and Medicine*, 93:66 – 74.

[Campanella et al., 2003] Campanella, J. J., Bitincka, L., et Smalley, J. (2003). **Matgat: an application that generates similarity/identity matrices using protein or dna sequences**. *BMC bioinformatics*, 4(1):1–4.

[Campello et al., 2013] Campello, R. J., Moulavi, D., et Sander, J. (2013). **Density-based clustering based on hierarchical density estimates**. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer.

[Celeux et al., 2001] Celeux, G., Chrétien, S., Forbes, F., et Mkhadri, A. (2001). **A component-wise em algorithm for mixtures**. *Journal of Computational and Graphical Statistics*, 10(4):697–712.

[Chen et al., 2013] Chen, W., Zhang, C. K., Cheng, Y., Zhang, S., et Zhao, H. (2013). **A comparison of methods for clustering 16s rrna sequences into otus**. *PloS one*, 8(8):e70837.

[Chrétien et al., 1998] Chrétien, S., et Hero, A. (1998). **Acceleration of the em algorithm via proximal point iterations**. In *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No. 98CH36252)*, page 444. IEEE.

[Chrétien et al., 2000] Chrétien, S., et Hero, A. O. (2000). **Kullback proximal algorithms for maximum-likelihood estimation**. *IEEE transactions on information theory*, 46(5):1800–1810.

[Dall'Amico et al., 2019a] Dall'Amico, L., Couillet, R., et Tremblay, N. (2019a). **Optimized deformed laplacian for spectrum-based community detection in sparse heterogeneous graphs**. *arXiv preprint arXiv:1901.09715*.

[Dall'Amico et al., 2019b] Dall'Amico, L., Couillet, R., et Tremblay, N. (2019b). **Revisiting the bethe-hessian: improved community detection in sparse heterogeneous graphs**. In *Advances in Neural Information Processing Systems*, pages 4037–4047.

[Demmel, 1991] Demmel, J. (1991). **Lapack: A portable linear algebra library for high-performance computers**. *Concurrency: Practice and Experience*, 3(6):655–666.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., et Rubin, D. B. (1977). **Maximum likelihood from incomplete data via the em algorithm**. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[Deng et al., 2006] Deng, X., Li, E., Shan, J., et Chen, W. (2006). **Parallel implementation and performance characterization of muscle**. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 7 pp.–.

[Duffy et al., 2008] Duffy, S., Shackelton, L. A., et Holmes, E. C. (2008). **Rates of evolutionary change in viruses: patterns and determinants**. *Nature Reviews Genetics*, 9(4):267.

[Edgar, 2004] Edgar, R. C. (2004). **Muscle: multiple sequence alignment with high accuracy and high throughput**. *Nucleic acids research*, 32(5):1792–1797.

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

[Fourment et al., 2008] Fourment, M., et Gillings, M. R. (2008). **A comparison of common programming languages used in bioinformatics**. *BMC Bioinformatics*, 9(1):82.

[Genovese et al., 2013] Genovese, M., et Napoli, E. (2013). **Asic and fpga implementation of the gaussian mixture model algorithm for real-time segmentation of high definition video**. *IEEE Transactions on very large scale integration (VLSI) systems*, 22(3):537–547.

[Ghodsi et al., 2011] Ghodsi, M., Liu, B., et Pop, M. (2011). **Dnaclust: accurate and efficient clustering of phylogenetic marker genes**. *BMC Bioinformatics*, 12(1):271.

[Girgis et al., 2018] Girgis, H. Z., James, B. T., et Luczak, B. B. (2018). **MeShClust: an intelligent tool for clustering DNA sequences**. *Nucleic Acids Research*, 46(14):e83–e83.

[Girgis et al., 2021] Girgis, H. Z., James, B. T., et Luczak, B. B. (2021). **Identity: rapid alignment-free prediction of sequence alignment identity scores using self-supervised general linear models**. *NAR genomics and bioinformatics*, 3(1):lqab001.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et others (2010). **Eigen v3**. http://eigen.tuxfamily.org.

[Guindon et al., 2009] Guindon, S., Delsuc, F., Dufayard, J.-F., et Gascuel, O. (2009). **Estimating maximum likelihood phylogenies with phyml**. In *Bioinformatics for DNA sequence analysis*, pages 113–137. Springer.

[Guindon et al., 2010] Guindon, S., Dufayard, J.-F., Lefort, V., Anisimova, M., Hordijk, W., et Gascuel, O. (2010). **New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0**. *Systematic biology*, 59(3):307–321.

[Gullberg et al., 2011] Gullberg, E., Cao, S., Berg, O. G., Ilbäck, C., Sandegren, L., Hughes, D., et Andersson, D. I. (2011). **Selection of resistant bacteria at very low antibiotic concentrations**. *PLoS pathogens*, 7(7):e1002158.

[Guo et al., 2012] Guo, C., Fu, H., et Luk, W. (2012). **A fully-pipelined expectation-maximization engine for gaussian mixture models**. In *2012 International Conference on Field-Programmable Technology*, pages 182–189. IEEE.

[Guyeux et al., 2019] Guyeux, C., Chrétien, S., Bou Tayeh, G., Demerjian, J., et Bahi, J. (2019). **Introducing and comparing recent clustering methods for massive data management in the internet of things**. *Journal of Sensor and Actuator Networks*, 8(4):56.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., et Friedman, J. (2001). **The elements of statistical learning, ser**.

[Hu et al., 2004] Hu, X., et Yoo, I. (2004). **Cluster ensemble and its applications in gene expression analysis**. In *Proceedings of the second conference on Asia-Pacific bioinformatics-Volume 29*, pages 297–302. Australian Computer Society, Inc.

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

[Katoh et al., 2013] Katoh, K., et Standley, D. M. (2013). **Mafft multiple sequence alignment software version 7: improvements in performance and usability**. *Molecular biology and evolution*, 30(4):772–780.

[Khan et al., 2014] Khan, K., Rehman, S. U., Aziz, K., Fong, S., et Sarasvady, S. (2014). **Dbscan: Past, present and future**. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE.

[Khurana et al., 2007] Khurana, S., Krementsov, D. N., de Parseval, A., Elder, J. H., Foti, M., et Thali, M. (2007). **Human immunodeficiency virus type 1 and influenza virus exit via different membrane microdomains**. *Journal of virology*, 81(22):12630–12640.

[Kulis et al., 2011] Kulis, B., et Jordan, M. I. (2011). **Revisiting k-means: New algorithms via bayesian nonparametrics**. *arXiv preprint arXiv:1111.0352*.

[Lang et al., 2008] Lang, G. I., et Murray, A. W. (2008). **Estimating the per-base-pair mutation rate in the yeast saccharomyces cerevisiae**. *Genetics*, 178(1):67–82.

[Langone et al., 2011] Langone, R., Alzate, C., et Suykens, J. A. (2011). **Modularity-based model selection for kernel spectral clustering**. In *The 2011 International Joint Conference on Neural Networks*, pages 1849–1856. IEEE.

[Larking et al., 2007] Larking, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, G., McWilliam, H., Valentin, F., Wallace, I., Wilm, A., Lopez, R., et others (2007). **Clustalw and clustalx version 2**. *Bioinformatics*, 23(21):2947–8.

[Lefort et al., 2017] Lefort, V., Longueville, J.-E., et Gascuel, O. (2017). **Sms: smart model selection in phyml**. *Molecular biology and evolution*, 34(9):2422–2424.

[Li, 2003] Li, K.-B. (2003). **Clustalw-mpi: Clustalw analysis using distributed and parallel computing**. *Bioinformatics*, 19(12):1585–1586.

[Li et al., 2006] Li, W., et Godzik, A. (2006). **Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences**. *Bioinformatics*, 22(13):1658–1659.

[Liakos et al., 2018] Liakos, K. G., Busato, P., Moshou, D., Pearson, S., et Bochtis, D. (2018). **Machine learning in agriculture: A review**. *Sensors*, 18(8):2674.

[Likas et al., 2003] Likas, A., Vlassis, N., et Verbeek, J. J. (2003). **The global k-means clustering algorithm**. *Pattern recognition*, 36(2):451–461.

[Lim et al., 2016] Lim, C., Takahashi, E., Hongsuwan, M., Wuthiekanun, V., Thamlikitkul, V., Hinjoy, S., Day, N. P., Peacock, S. J., et Limmathurotsakul, D. (2016). **Epidemiology and burden of multidrug-resistant bacterial infection in a developing country**. *Elife*, 5:e18082.

[Malzer et al., 2020] Malzer, C., et Baum, M. (2020). **A hybrid approach to hierarchical density-based cluster selection**. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 223–228. IEEE.

[Matar et al., 2021] Matar, J., ElKhoury, H., Charr, J.-C., Guyeux, C., et Chrétien, S. (2021). **Optimized spectral clustering methods for potentially divergent biological sequences**. Preprint on https://www.researchgate.net/project/Spectral-clustering-for-biological-sequences.

[Matar et al., 2019] Matar, J., Khoury, H. E., Charr, J.-C., Guyeux, C., et Chrétien, S. (2019). **Spclust: Towards a fast and reliable clustering for potentially divergent biological sequences**. *Computers in biology and medicine*, 114:103439.

[Matias Rodrigues et al., 2013] Matias Rodrigues, J. F., et von Mering, C. (2013). **Hpc-clust: distributed hierarchical clustering for large sets of nucleotide sequences**. *Bioinformatics*, 30(2):287–288.

[McGregor et al., 2004] McGregor, A., Hall, M., Lorier, P., et Brunskill, J. (2004). **Flow clustering using machine learning techniques**. In *International workshop on passive and active network measurement*, pages 205–214. Springer.

[McLachlan et al., 2007] McLachlan, G. J., et Krishnan, T. (2007). **The EM algorithm and extensions**, volume 382. John Wiley & Sons.

[McLachlan et al., 2004] McLachlan, G. J., et Peel, D. (2004). **Finite mixture models**. John Wiley & Sons.

[Mercier et al., 2013] Mercier, C., Boyer, F., Bonin, A., et Coissac, E. (2013). **Sumatra and sumaclust: fast and exact comparison and clustering of sequences**. In *Programs and Abstracts of the SeqBio 2013 workshop. Abstract*, pages 27–29. Citeseer.

[Mirarab et al., 2015] Mirarab, S., Nguyen, N., Guo, S., Wang, L.-S., Kim, J., et Warnow, T. (2015). **Pasta: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences**. *Journal of Computational Biology*, 22(5):377–386.

[Mirkin, 2012] Mirkin, B. (2012). **Clustering: A data recovery approach, vol. 19**.

[Müller et al., 2011] Müller, M., Parkhurst, D. L., et Charlton, S. R. (2011). **Programming phreeqc calculations with c++ and python a comparative study**. *EXCHANGE*, 1(40):632–636.

[Murtagh et al., 2012] Murtagh, F., et Contreras, P. (2012). **Algorithms for hierarchical clustering: an overview**. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97.

[Myers et al., 1988] Myers, E. W., et Miller, W. (1988). **Optimal alignments in linear space**. *Bioinformatics*, 4(1):11–17.

[Notredame et al., 2000] Notredame, C., Higgins, D. G., et Heringa, J. (2000). **T-coffee: a novel method for fast and accurate multiple sequence alignment1**. *Journal of molecular biology*, 302(1):205–217.

[Nuin et al., 2006] Nuin, P. A., Wang, Z., et Tillier, E. R. (2006). **The accuracy of several multiple sequence alignment programs for proteins**. *BMC bioinformatics*, 7(1):471.

[Oliphant, 2007] Oliphant, T. E. (2007). **Python for scientific computing**. *Computing in Science Engineering*, 9(3):10–20.

[Oliver et al., 2010] Oliver, A., et Mena, A. (2010). **Bacterial hypermutation in cystic fibrosis, not only for antibiotic resistance**. *Clinical Microbiology and Infection*, 16(7):798–808.

[Paaßen et al., 2018] Paaßen, B., Schulz, A., Hahne, J., et Hammer, B. (2018). **Expectation maximization transfer learning and its application for bionic hand prostheses**. *Neurocomputing*, 298:122–133.

[Paccanaro et al., 2006] Paccanaro, A., Casbon, J. A., et Saqi, M. A. (2006). **Spectral clustering of protein sequences**. *Nucleic acids research*, 34(5):1571–1580.

[Pearson, 2013] Pearson, W. R. (2013). **An introduction to sequence similarity ("homology") searching**. *Current protocols in bioinformatics*, 42(1):3–1.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., et Duchesnay, E. (2011). **Scikit-learn: Machine learning in Python**. *Journal of Machine Learning Research*, 12:2825–2830.

[Pentney et al., 2005] Pentney, W., et Meila, M. (2005). **Spectral clustering of biological sequence data**. In *AAAI*, volume 5, pages 845–850.

[Pipes et al., 2021] Pipes, L., et Nielsen, R. (2021). **Ancestralclust: Clustering of divergent nucleotide sequences by ancestral sequence reconstruction using phylogenetic trees**. *bioRxiv*.

[Ronquist et al., 2012] Ronquist, F., Teslenko, M., Van Der Mark, P., Ayres, D. L., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M. A., et Huelsenbeck, J. P. (2012). **Mrbayes 3.2: efficient bayesian phylogenetic inference and model choice across a large model space**. *Systematic biology*, 61(3):539–542.

[Saade et al., 2014] Saade, A., Krzakala, F., et Zdeborová, L. (2014). **Spectral clustering of graphs with the bethe hessian**. In *Advances in Neural Information Processing Systems*, pages 406–414.

[Sameh, 1971] Sameh, A. H. (1971). **On jacobi and jacobi-like algorithms for a parallel computer**. *Mathematics of computation*, 25(115):579–590.

[Sanderson et al., 2016] Sanderson, C., et Curtin, R. (2016). **Armadillo: a template-based c++ library for linear algebra**. *Journal of Open Source Software*, 1(2):26.

[Sanderson et al., 2017] Sanderson, C., et Curtin, R. (2017). **gmm_diag and gmm_full: C++ classes for multi-threaded gaussian mixture models and expectation-maximisation**. *Journal of Open Source Software*, 2(18):365.

[Santos et al., 2009] Santos, J. M., et Embrechts, M. (2009). **On the use of the adjusted rand index as a metric for evaluating supervised classification**. In *International conference on artificial neural networks*, pages 175–184. Springer.

[Schulte im Walde, 2003] Schulte im Walde, S. (2003). **Experiments on the Automatic Induction of German Semantic Verb Classes**. PhD thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart. Published as AIMS Report 9(2).

[Schütze et al., 2008] Schütze, H., Manning, C. D., et Raghavan, P. (2008). **Introduction to information retrieval**, volume 39. Cambridge University Press Cambridge.

[Shi et al., 2006] Shi, M., et Bermak, A. (2006). **An efficient digital vlsi implementation of gaussian mixture models-based classifier**. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(9):962–974.

[Somashekara et al., 2014] Somashekara, M., et Manjunatha, D. (2014). **Performance evaluation of spectral clustering algorithm using various clustering validity indices**. *International Journal of Electronics Communication and Computer Engineering*, 5(6):1274–1276.

[Sørensen et al., 2003] Sørensen, M., Autrup, H., Møller, P., Hertel, O., Jensen, S. S., Vinzents, P., Knudsen, L. E., et Loft, S. (2003). **Linking exposure to environmental pollutants with biological effects**. *Mutation Research/Reviews in Mutation Research*, 544(2):255–271.

[Šošić et al., 2017] Šošić, M., et Šikić, M. (2017). **Edlib: a c/c++ library for fast, exact sequence alignment using edit distance**. *Bioinformatics*, 33(9):1394–1395.

[Stamatakis, 2014] Stamatakis, A. (2014). **Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies**. *Bioinformatics*, 30(9):1312–1313.

[Thompson et al., 2002] Thompson, J. D., Gibson, T. J., et Higgins, D. G. (2002). **Multiple sequence alignment using clustalw and clustalx**. *Current Protocols in Bioinformatics*, 00(1):2.3.1–2.3.22.

[Ventola, 2015] Ventola, C. L. (2015). **The antibiotic resistance crisis: part 1: causes and threats**. *Pharmacy and Therapeutics*, 40(4):277.

[Von Luxburg, 2007] Von Luxburg, U. (2007). **A tutorial on spectral clustering**. *Statistics and computing*, 17(4):395–416.

[Vrieze, 2012] Vrieze, S. I. (2012). **Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic).** *Psychological methods*, 17(2):228.

[Wagner et al., 2007] Wagner, S., et Wagner, D. (2007). **Comparing clusterings: an overview**. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

[Wang et al., 2019] Wang, X., et Xu, Y. (2019). **An improved index for clustering validation based on silhouette index and calinski-harabasz index**. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052024. IOP Publishing.

[Wielgoss et al., 2013] Wielgoss, S., Barrick, J. E., Tenaillon, O., Wiser, M. J., Dittmar, W. J., Cruveiller, S., Chane-Woon-Ming, B., Médigue, C., Lenski, R. E., et Schneider, D. (2013). **Mutation rate dynamics in a bacterial population reflect tension between adaptation and genetic load**. *Proceedings of the National Academy of Sciences*, 110(1):222–227.

[Wilm et al., 2006] Wilm, A., Mainz, I., et Steger, G. (2006). **An enhanced rna alignment benchmark for sequence alignment programs**. *Algorithms for molecular biology*, 1(1):19.

[Wright, 2015] Wright, E. S. (2015). **Decipher: harnessing local sequence context to improve protein multiple sequence alignment**. *Bmc Bioinformatics*, 16(1):322.

[Wu, 1983] Wu, C. J. (1983). **On the convergence properties of the em algorithm**. *The Annals of statistics*, pages 95–103.

List of Figures

List of Tables

V

APPENDIX

CLUSTERS CONTENTS

C1	C2	C3	C4
H.sapiens_MC1R	H.sapiens_SH3BGRL3	H.sapiens_S100A6	G.gallus_S100A6
P.troglodytes_MC1R	C.lupus_SH3BGRL3	P.troglodytes_S100A6	H.sapiens_S100A8
C.lupus_MC1R	B.taurus_SH3BGRL3	M.mulatta_S100A6	M.mulatta_S100A8
B.taurus_MC1R	M.musculus_Sh3bgrl3	C.lupus_S100A6	C.lupus_S100A8
M.musculus_Mc1r	R.norvegicus_Sh3bgrl3	B.taurus_S100A6	B.taurus_S100A8
R.norvegicus_Mc1r		M.musculus_S100a6	M.musculus_S100a8
G.gallus_MC1R		R.norvegicus_S100a6	R.norvegicus_S100a8
D.rerio_mc1r		G.gallus_SH3BGRL3	H.sapiens_S100A1
		H.sapiens_FCER1G	P.troglodytes_S100A1
		P.troglodytes_FCER1G	M.mulatta_S100A1
		C.lupus_FCER1G	C.lupus_S100A1
		B.taurus_FCER1G	B.taurus_S100A1
		M.musculus_Fcer1g	M.musculus_S100a1
		R.norvegicus_Fcer1g	R.norvegicus_S100a1
			G.gallus_S100A1
			D.rerio_s100a1
			H.sapiens_S100A12
			P.troglodytes_S100A12
			C.lupus_S100A12
			B.taurus_S100A12

Table 1: Clustering - Real data genomic set 2

C1	C2	C3	C4
KU758869 ZIKA	CY083917 H1N1 PB2	CY021939 H2N2 PB1	CY021027 H2N2 PB1
KU312313 ZIKA	CY063613 H1N1 PB2	CY020323 H2N2 PB1	AY210016 H2N2 PB1
KU758873 ZIKA	CY083782 H1N1 PB2	CY022019 H2N2 PB1	CY020419 H2N2 PB1
KU758868 ZIKA	CY073732 H1N1 PB2	CY021811 H2N2 PB1	
KU312314 ZIKA	CY062698 H1N1 PB2	CY021795 H2N2 PB1	
KU758872 ZIKA	CY062706 H1N1 PB2		
KU758876 ZIKA			
KU758871 ZIKA			
KU758870 ZIKA			
KU758875 ZIKA			

Table 2: Clustering - Real data genomic set 4

C1	C2	C3	C4
AML81020 ZIKA	ADX98969 H1N1 PB2	ABQ01363 H2N2 PB1	ABO52255 H2N2 PB1
ALX35660 ZIKA	ADH01967 H1N1 PB2	ABO38106 H2N2 PB1	AAO46332 H2N2 PB1
AML81024 ZIKA	ADX98798 H1N1 PB2	ABQ44468 H2N2 PB1	ABO38742 H2N2 PB1
AML81019 ZIKA	ADN05235 H1N1 PB2	ABP49467 H2N2 PB1	
ALX35661 ZIKA	ADG42162 H1N1 PB2	ABP49445 H2N2 PB1	
AML81023 ZIKA	ADG42172 H1N1 PB2		
AML81027 ZIKA			
AML81022 ZIKA			
AML81021 ZIKA			
AML81026 ZIKA			

Table 3: Clustering - Real data protein set 4

C1	C2	C3	C4
H.sapiens_MC1R	H.sapiens_SH3BGRL3	H.sapiens_S100A6	H.sapiens_S100A8
P.troglodytes_MC1R	C.lupus_SH3BGRL3	P.troglodytes_S100A6	M.mulatta_S100A8
C.lupus_MC1R	B.taurus_SH3BGRL3	M.mulatta_S100A6	C.lupus_S100A8
B.taurus_MC1R	M.musculus_Sh3bgrl3	C.lupus_S100A6	B.taurus_S100A8
M.musculus_Mc1r	R.norvegicus_Sh3bgrl3	B.taurus_S100A6	M.musculus_S100a8
R.norvegicus_Mc1r		M.musculus_S100a6	
G.gallus_MC1R		R.norvegicus_S100a6	
D.rerio_mc1r		G.gallus_S100A6	
		G.gallus_SH3BGRL3	
		R.norvegicus_S100a8	
		H.sapiens_S100A1	
		P.troglodytes_S100A1	
		M.mulatta_S100A1	
		C.lupus_S100A1	
		B.taurus_S100A1	
		M.musculus_S100a1	
		R.norvegicus_S100a1	
		G.gallus_S100A1	
		D.rerio_s100a1	
		H.sapiens_FCER1G	
		P.troglodytes_FCER1G	
		C.lupus_FCER1G	
		B.taurus_FCER1G	
		M.musculus_Fcer1g	
		R.norvegicus_Fcer1g	
		H.sapiens_S100A12	
		P.troglodytes_S100A12	
		C.lupus_S100A12	
		B.taurus_S100A12	

Table 4: Clustering - Real data genomic set 2 - Fast alignment

SEQUENCES REFERENCES

Source	Accession ID	Collection date	Lineage	Originating organism
GISAID	EPI_ISL_1057903	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057905	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057906	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057911	2021-02-08	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057912	2021-02-08	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057913	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057915	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057917	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057918	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057919	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057920	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057921	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057924	2021-02-09	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057925	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057926	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057927	2021-02-08	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057928	2021-02-08	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057930	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057935	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057935	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057937	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057938	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057940	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057941	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057943	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057944	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057946	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057947	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057948	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057952	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057953	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057954	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057955	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057959	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057962	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057963	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057964	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057967	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057968	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057969	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057971	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057973	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057974	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057975	2021-02-09	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_1057976	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057977	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1057978	2021-02-13	B.1.1.7	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_1058003	2021-02-11	B.1.1.7	Lighthouse Lab in Alderley Park

Table 5: SARS-COV2 complete genome samples collected in the UK.

Source	Accession ID	Collection date	Lineage	Originating organism
GISAID	EPI_ISL_1048460	2021-01-04	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048461	2021-01-04	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048462	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048463	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048466	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048469	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048470	2021-01-09	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048471	2021-01-08	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048473	2021-01-09	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048474	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048475	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048482	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048487	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048489	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048490	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048491	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048492	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048493	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048497	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048500	2021-01-12	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048503	2021-01-16	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048504	2021-01-17	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048507	2021-01-15	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048512	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048514	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048515	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048516	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048517	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048519	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048523	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048524	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048526	2021-01-19	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048527	2021-01-19	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048528	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048529	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048532	2021-01-18	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048533	2021-01-19	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048534	2021-01-20	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048535	2021-01-11	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048541	2021-01-31	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048547	2021-01-30	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048550	2021-01-31	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048553	2021-02-02	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048554	2021-02-02	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048557	2021-01-31	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048558	2021-01-29	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048559	2021-01-26	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048560	2021-01-29	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048562	2021-02-01	B.1.351	National Health Laboratory Service
GISAID	EPI_ISL_1048568	2021-01-29	B.1.351	National Health Laboratory Service

Table 6: SARS-COV2 complete genome samples collected in South Africa.

Source	Accession ID	Collection date	Lineage	Originating organism
GISAID	EPI_ISL_855634	2021-01-11	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_928281	2021-01-11	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_954180	2021-01-25	B.1.525	Vall d'Hebron Institut de Recerca - Spain
GISAID	EPI_ISL_973299	2021-01-25	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_973450	2021-01-25	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1009673	2021-01-27	B.1.525	CHI Andre Gregoire - France
GISAID	EPI_ISL_1022477	2021-02-08	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1023333	2021-02-08	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1023711	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1023945	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1023997	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024134	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024295	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024510	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024746	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024750	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1024816	2021-02-01	B.1.525	Statens Serum Institut - Denmark
GISAID	EPI_ISL_1036755	2021-02-03	B.1.525	Universitaria di Bologna - Italy
GISAID	EPI_ISL_1049122	2021-01-28	B.1.525	Rega Institute - Belgium
GISAID	EPI_ISL_1049260	2021-02-03	B.1.525	Universitaria di Bologna - Italy
GISAID	EPI_ISL_1059438	2021-01-22	B.1.525	Viollier AG - Switzerland

Table 7: SARS-COV2 complete genome samples collected in Europe.

Source	Org. seq. acc. ID	Collection date	Lineage	Originating organism
GISAID	EPI_ISL_664324	2020-11-11	B.1.177	Department of Pathology
GISAID	EPI_ISL_608089	2020-10-14	B.1.177	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_535182	2020-08-31	B.1.177	West of Scotland Specialist Virology Centre
GISAID	EPI_ISL_588703	2020-09-24	B.1.177	Lighthouse Lab in Glasgow
GISAID	EPI_ISL_593810	2020-10-04	B.1.177	Respiratory Virus Unit
GISAID	EPI_ISL_727954	2020-12-17	B.1.177	Virology Department
GISAID	EPI_ISL_612112	2020-10-02	B.1.177	Liverpool Clinical Laboratories
GISAID	EPI_ISL_627978	2020-10-16	B.1.177	Wales Specialist Virology Centre Sequencing lab
GISAID	EPI_ISL_551308	2020-06-10	B.1.1	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_597366	2020-10-07	B.1.1	Lighthouse Lab in Cambridge
GISAID	EPI_ISL_464300	2020-03-03	B.1.1	Respiratory Virus Unit
GISAID	EPI_ISL_874479	2021-01-12	B.1.1.7	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_835855	2021-01-03	B.1.1.7	Lighthouse Lab in Milton Keynes
GISAID	EPI_ISL_881967	2021-01-10	B.1.1.7	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_811123	2021-01-08	B.1.1.7	Respiratory Virus Unit
GISAID	EPI_ISL_863167	2021-01-11	B.1.1.7	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_874727	2021-01-12	B.1.1.7	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_867779	2021-01-07	B.1.1.7	Wales Specialist Virology Centre
GISAID	EPI_ISL_863383	2021-01-12	B.1.1.7	Lighthouse Lab in Alderley Park
GISAID	EPI_ISL_846090	2021-01-09	B.1.1.7	Lighthouse Lab in Cambridge

Table 8: Spike protein from SARS-COV2 samples collected in the UK.

Source	Accession ID	Collection date	Sample origin	Originating organism
NCBI	AYV99817.1	2018-10-17	USA	University of North Carolina at Chapel Hill - USA
NCBI	ABA02260.1	2003-04-21	China	Zhejiang CDC - China
NCBI	AAR86775.1		China	Key Lab of Medical Molecular Virology - China
NCBI	AAT74874.1		China	Institute of Microbiology - China
NCBI	BAE93401.1			Research Institute of Microbial Diseases - Japan

Table 9: Spike protein from SARS-COV samples.

Source	Accession ID	Collection date	Sample origin	Originating organism
NCBI	QBM11748.1	2017-08-24	Ethiopia	National Institute of Infectious Diseases - Japan
NCBI	QBM11737.1	2017-08-24	Ethiopia	National Institute of Infectious Diseases - Japan
NCBI	AHX00711.1	2013-12-30	Saudi Arabia	The University of Hong Kong - Hong Kong

Table 10: Spike protein from MERS samples.

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Optimizing machine learning techniques for genomics clustering

Keywords: Biological sequences clustering, Genomics, Laplacian Eigenmaps, Gaussian Mixture Model, Parallel computation, Spectral clustering, Clustering quality analysis.

Abstract:

In the field of bioinformatics, clustering recently appeared to be a very efficient technique for sequence analysis. While greedy and hierarchical algorithms are used in the majority of the available tools, spectral clustering was recently introduced as a new stakeholder in this field. Spectral clustering is an efficient technique for well separated sequence clustering and GMM's are often able to cluster overlapping groups given an adequately designed embedding. Yet, the traditional clustering tools present many drawbacks such as the need for non-obvious parameters and the lack of optimization for handling potentially divergent sequences. Moreover, a newly introduced technique that targets the clustering of potentially divergent sequences, was only experimented on a single dataset. Furthermore, the performance of several well-known clustering techniques is not assessed in the field of clustering biological sequences.
This dissertation mainly focuses on validating and optimizing novel techniques for clustering biological sequences, which present unknown and possibly high levels of divergence. To do so, two main axes have been considered, namely, the clustering techniques, and the processing speed. In the first axis, novel clustering techniques have been proposed and evaluated in the sequence clustering field, to solve the limitations imposed by the traditional techniques. While the second axis tackled the speed optimization of the valid techniques, by offering more efficient implementation schemes such as substituting external modules, porting some modules to lower level programming languages, and using parallel computation.

Titre : Optimisation des techniques d'apprentissage automatique pour le clustering génomique

Mots-clés : Clustering de séquences biologiques, Génomique, Eigenmaps laplaciennes, Modèle de mélange gaussien, Calcul parallèle, Clustering spectral, Analyse de qualité de clustering.

Résumé :

Dans le domaine de la bioinformatique, le clustering est récemment apparu comme une technique très efficace pour l'analyse des séquences. Alors que des algorithmes gloutons et hiérarchiques sont utilisés dans la majorité des outils disponibles, le clustering spectral a récemment été introduit comme un nouvel acteur dans ce domaine. Le clustering spectral est une technique efficace pour le clustering de séquences bien séparées et les GMM sont souvent capables de partitionner des groupes qui intersectent, étant donné une intégration adéquatement conçue. Pourtant, les outils de clustering traditionnels présentent de nombreux obstacles tels que le besoin de paramètres non évidents et le manque d'optimisation pour gérer des séquences potentiellement divergentes. De plus, une technique nouvellement introduite, qui cible le regroupement de séquences potentiellement divergentes, n'a été expérimentée que sur un seul ensemble de données. De plus, les performances de plusieurs techniques de clustering, bien connues, ne sont pas évaluées dans le domaine du clustering de séquences biologiques.
Cette thèse se focalise principalement sur la validation et l'optimisation de nouvelles techniques de regroupement de séquences biologiques, qui présentent des niveaux de divergence inconnus et éventuellement élevés. Afin de realiser ce but, deux axes principaux ont été considérés, à savoir, les techniques de clustering et la vitesse de traitement. Dans le premier axe, de nouvelles techniques de clustering ont été proposées et évaluées dans le domaine du clustering de séquences, pour résoudre les limitations imposées par les techniques traditionnelles. Alors que le deuxième axe s'est attaqué à l'optimisation de la vitesse des techniques valides, en proposant des schémas de mise en œuvre plus efficaces, tels que la substitution de modules externes, le recodage de certaines modules dans des langages de programmation bas-niveau et l'utilisation de la technique de calcul parallèle.

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France