Thesis for obtaining the title of
Doctor of Natural Science
of the Faculty of Natural Science and Technology I
of Saarland University

Active Transitivity Clustering of Large-Scale Biomedical Datasets

by

Dipl.-Inf. Richard Röttger

Saarbrücken
February, 2014

Day of Colloquium:	28 / 05 / 2014
Dean of the Faculty:	Univ.-Prof. Dr. Markus Bläser
Chair of the Committee:	Prof. Dr.-Ing. Thorsten Herfet
First Reviewer:	Prof. Dr. Jan Baumbach
Second Reviewer:	Prof. Dr. Dr. Thomas Lengauer
Third Reviewer:	Prof. Dr. Nataša Pržulj
Academic Assistant:	Dr. Marc Hellmuth

Executive Summary

Clustering is a popular computational approach for partitioning data sets into groups of objects that share common traits. Due to recent advances in wet-lab technology, the amount of available biological data grows exponentially and increasingly poses problems in terms of computational complexity for current clustering approaches. In this thesis, we introduce two novel approaches, **TransClustMV** and **ActiveTransClust**, that enable the handling of large scale datasets by reducing the amount of required information drastically by means of exploiting missing values.

Furthermore, there exists a plethora of different clustering tools and standards making it very difficult for researchers to choose the correct methods for a given problem. In order to clarify this multifarious field, we developed **ClustEval** which streamlines the clustering process and enables practitioners conducting large-scale cluster analyses in a standardized and bias-free manner.

We conclude the thesis by demonstrating the power of clustering tools and the need for the previously developed methods by conducting real-world analyses. We transferred the regulatory network of *E. coli* K-12 to pathogenic EHEC organisms based on evolutionary conservation therefore avoiding tedious and potentially dangerous wet-lab experiments. In another example, we identify pathogenicity specific core genomes of actinobacteria in order to identify potential drug targets.

Kurzzusammenfassung

Clustering ist ein populärer Ansatz um Datensätze in Gruppen ähnlicher Objekte zu partitionieren. Nicht zuletzt aufgrund der jüngsten Fortschritte in der Labortechnik wächst die Menge der biologischen Daten exponentiell und stellt zunehmend ein Problem für heutige Clusteralgorithmen dar. Im Rahmen dieser Arbeit stellen wir zwei neue Ansätze, **TransClustMV** und **ActiveTransClust**, vor die auch das Bearbeiten sehr großer Datensätze ermöglichen, indem sie den Umfang der benötigten Informationen drastisch reduzieren da fehlende Werte kompensiert werden können.

Allein die schiere Vielfalt der vorhanden Cluster-Methoden und Standards stellt den Anwender darüber hinaus vor das Problem, den am besten geeigneten Algorithmus für das vorliegende Problem zu wählen. **ClustEval** wurde mit dem Ziel entwickelt, diese Unübersichtlichkeit zu beseitigen und gleichzeitig die Clusteranalyse zu vereinheitlichen und zu automatisieren um auch aufwendige Clusteranalysen zu realisieren.

Abschließend demonstrieren wir die Nützlichkeit von Clustering anhand von realen Anwendungsfällen die darüber hinaus auch den Bedarf der zuvor entwickelten Methoden aufzeigen. Wir haben das genregulatorische Netzwerk von *E. coli* K-12 ohne langwierige und potentiell gefährliche Laborarbeit auf pathogene EHEC Stämme übertragen. In einem weiteren Beispiel bestimmen wir das pathogenitätsspeziefische „Kerngenom“ von Actinobakterien um potenzielle Angriffspunkte für Medikamente zu identifizieren.

Abstract

Motivation

Currently, researchers conducting cluster analyses face a set of methodological and practical problems. We identify the following three most critical objectives:

Missing Values
Especially in the post-genome area, a tremendous growth of available biological datasets of ever increasing precision and quality can be observed. As welcome as this growing amount of available biological datasets is, this trend also poses a multitude of problems when applied to clustering algorithms. Particularly problematic is the calculation of pairwise similarities between all objects, a necessary prerequisite for a cluster analysis. It appears to be more and more the actual bottleneck, especially when applying complicated similarity measures. The existence of this bottleneck has been widely neglected so far but limits the research community in conducting clustering analyses on a large scale. Thus, sophisticated methods have to be developed which also enable the usage of complex similarity functions for massive datasets.

Unifying Clustering
As the second problem for the practitioner we identified the plethora of existing clustering algorithms being a huge obstacle. Every newly proposed method is compared only to a selected handful of already existing tools on a possibly biased selection of datasets. Furthermore, almost all clustering tools use their very own input and output format which renders the conduction of large comparative clustering studies a very exhausting and error-prone process.

Practical Issues
Even in cases were the researcher has identified a feasible clustering tool able to cope with the dataset size, the actual execution of a cluster analysis still poses several problems. For most problems, reliable gold standards do not exist; thus, finding a meaningful and unbiased density parameter poses a problem. Furthermore, the results of a cluster analysis need to be integrated into entire pipelines and combined with other datasets. All this impacts the clustering process and guidelines are needed.

Results

In the framework of this thesis, we developed a solution for the aforementioned objectives:

Missing Values
We show that most of the calculated pairwise similarities only carry redundant infor-

mation, and neglecting them barely influences the clustering result. Our main contribution is the development of methods enabling the existing and widely accepted clustering tool TransClust to strategically exploit this fact and produce high quality cluster results while using only a small fraction of the pairwise similarities. We introduced two new approaches, TransClustMV and ActiveTransClust. With that, we were able to reduce the required time for the calculation of the similarity file drastically, and thus enable the usage of computationally intensive similarity functions on massive datasets.

Unifying Clustering
With ClustEval, we introduce an integrated clustering framework, assisting the user in all steps of cluster analyses, from data preprocessing and parameter optimization to evaluating the reported clusters. The flexibility of the framework allows convenient extension with new tools, datasets, and quality measures. Furthermore, the layman is able to inform himself about the different clustering tools and their performance upon different types of datasets on an easy to use website.

Practical Issues
We demonstrate the integration of our clustering tool into an entire biological framework in a real world application. We successfully transferred the regulatory network of *E. coli* K-12 to several pathogenic EHEC strains based on evolutionary conservation avoiding tedious and potentially dangerous wet-lab experiments. In a second example, we identified the pathogenicity core-genome of numerous bacteria of the phylum actinobacterium. In this study, we also developed a reliable method for determining a meaningful threshold for protein homology detection without gold standard using only intrinsic information of the dataset.

In conclusion, we developed three novel tools and frameworks, **TransClustMV**, **ActiveTransClust**, and **ClustEval**, in order to unify cluster analyses in bioinformatics and meet today's requirements for state of the art clustering approaches.

Acknowledgements

First of all, I want to express my gratitude to my supervisor Prof. Jan Baumbach and Prof. Thomas Lengauer for not only granting me the possibility to work on such an interesting and challenging topic but also their support, advice and motivation throughout the entire thesis and setting up such a fruitful environment allowing my own ideas to prosper and maturate. I dedicate special thanks to Prof. Jan Baumbach for his tireless commitment to setting my yet young scientific career on track and for his advice and friendship outside of the office.

I also want to thank the International Max Planck Research School for granting my fellowship, the Center for Bioinformatics for providing me office space and material, the Cluster of Excellence on Multimodal Computing and Interaction and the University of the Saarland for financing my traveling. In all institutes, I would like to thank the supporting staff and IT helpdesk for their always fast and exemplary help and support, namely, Sabine Nermerich, Polina Quaranta, Jennifer Gerling and Dirk Raufer.

Of course, all colleagues of the Computational Systems Biology group, including my office mate Anne-Christin, Rashid, Peng, Nic, and Josch, have earned my gratefulness for their kindness, input, valuable feedback and inspiring discussions. Special thanks is dedicated to my student assistants Christoph L., Christian, Alexander, Prabhav, and Christoph K. for their great work and dedication to the projects. I also want to thank my colleagues in Brasil, Prof. Vasco Azevedo, Vini, Eudes and Julio, and in Newcastle, Prof. Anil Wipat and Goksel for their collaboration, successful publications and valuable input.

Outside the University, I owe countless people my thank-yous and deepest respect. I especially want to thank Laura and Fabian, for the tiresome and tedious task of proofreading my thesis, and providing inspiring input and suggestions for improving my work. I want to dedicate the last sentence of my acknowledgements to my loving, caring and supportive parents Margit and Walter and my brother Michael for being such a great support not only during the thesis, but during my entire life.

Contents

1. Introduction

1.1. Motivation

Clustering is an *unsupervised* machine-learning technique [7, 127, 47, 60] and describes the task of grouping objects in a given dataset in such a way that all objects in a group (called a *cluster*) are more similar to each other than to objects of different clusters [60]. The set of all clusters forms a *clustering*. In this context, unsupervised means that no *a priori* knowledge about the classification of the objects is used [7]. Clustering is used in various disciplines, e.g., information retrieval [104], economics and marketing [27], astronomy [9, 26], archeology [77], and many others (most examples and citations were taken from [47]). In the field of computational biology, clustering is applied to various different problems, e.g., unraveling protein-protein interactions [22], gene expression data analysis [4], and many other applications.

In this work, we distinguish between three different kinds of clustering [120, 90, 127]:

Partitional Clustering (sometimes also called *disjoint* or *crisp* clustering): Each object is assigned to exactly one cluster [120, 90, 127].

Overlapping/Fuzzy Clustering: Each object may be assigned to more than one cluster [120, 90, 127].

Hierarchical Clustering: In this case, the algorithm reports hierarchically-nested sets of clusters, ranging from one cluster containing all objects to a set of clusters each containing exactly one object [55].

Refer to Subsection 2.1.2 on page 24 for formal definitions. In the course of this thesis, we mainly discuss partitional clusterings. A central aspect of every clustering task is the definition of the relationship between the given objects [60]. This is achieved by the definition of a:

Proximity Measure A proximity measure is a function assigning each object pair either a similarity (similarity measure) or a distance (distance measure).

This proximity information is often represented in a *proximity matrix* (or *similarity/distance matrix*) which serves as the input (in the form of a *similarity/distance file*) of a clustering method [60]. Furthermore, every partitional clustering method requires at least one parameter [57].

Parameter A partitional clustering algorithm takes at least one parameter as an input and attempts to report the best clustering for the given dataset with respect to the parameter [57].

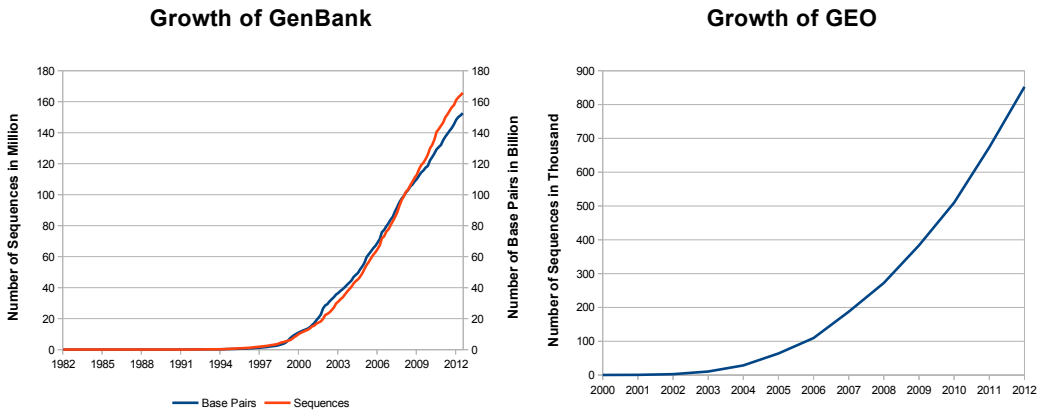


Figure 1.1.: Growth of the GenBank database (left) and the GenExpressionOmnibus (right). Both databases have shown the same exponential-like growth since their existence.

Clustering tools may require a single parameter (then often called *density parameter* or *threshold*) or a parameter set. The parameter k in the well-known K-means approach defining the number of desired clusters may serve as an example for such a parameter. Refer to Section 2.4 on page 39 for an overview of the clustering algorithms and their according parameters.

Generally, the aim of a cluster analysis is unraveling the hidden structure of datasets which are too large for a manual inspection, thus retrieving information more efficiently [47]. In the light of the ever growing amount of available biological data, clustering becomes increasingly popular [127]. As an example, GenBank [18], hosted on the websites of the NCBI [86], contains more than 165 million sequences resulting in more than 152 billion base-pairs[1], doubling its contents every 18 months. Figure 1.1 depicts the growth of the GenBank since its existence. In total, the full genome sequences of 200 eukaryotes (including 44 mammals), 2,599 prokaryotes (including 2,442 bacteria and 157 archaea) and 3,845 viruses are ready for download from the websites of the NCBI[2]. The same growth can also be observed in other databases, like gene expression studies collected at the GenExpressionOmnibus (GEO) [40, 12] holding currently almost one million samples[3]. Figure 1.1 also depicts the growth of GEO. These two measures may serve as an example for an ongoing observation: The development of new wet-lab techniques foster further speed-up of the database growth. For example, emerging next-generation sequencing technologies have drastically reduced the cost for DNA sequencing over the last years, allowing for entirely new possibilities of research [80, 105].

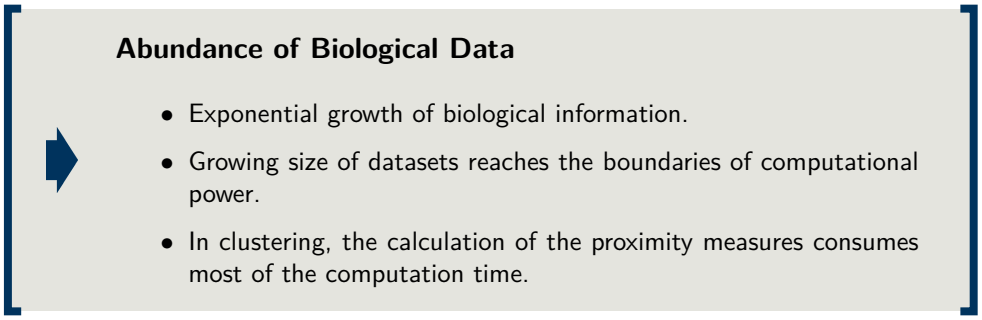
[1]These numbers are taken from the GenBank growth statistics document of June 15th, 2013, Subsection 2.2.8, "Growth of GenBank". The up-to-date version can be found at ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt.

[2]The numbers were taken from the NCBI genome browser filtering on complete projects only. The numbers reflect the state as of 6th August 2013. http://www.ncbi.nlm.nih.gov/genome/browse/

[3]Statistics taken from http://www.ncbi.nlm.nih.gov/geo/summary/ on August 6th, 2013.

On the one hand, data abundance is a blessing for systems biology, as it enables new perspectives for research. On the other hand, computational power of current-day computers is not always sufficient to cope with this increasing amount of available data [67, 61]. Frequently, the calculation of the the $\binom{N}{2}$ pairwise proximities for N objects consumes the main share of the computation time of a cluster analysis [29, 67, 61]. In order to enable clustering tools to cope with millions of objects, specialized, single-purpose clustering tools have been developed. These utilize a case-specific approximate similarity function, e.g., in [61, 71].

In contrast to these single-purpose tools, we aim to enable researchers to utilize the general-purpose clustering tool TransClust [122, 123, 125] using almost any proximity function. Therefore, we have to reduce the required computational time spent in calculating the pairwise similarities. With *TransClustMV* and *ActiveTransClust*, we introduce two novel clustering approaches which only require parts of the proximity matrix for clustering. This characteristic results in a reduction in both the size of the proximity file and the runtime used to calculate the proximity matrix. Refer to Section 1.2 on page 17 where we clarify the objectives of this thesis.



Even in cases where the size of the dataset does not pose a problem in terms of computational time, a cluster analysis is still error-prone and complicated. In order to apply a clustering algorithm to a certain dataset, the practitioner has to overcome several different obstacles. One challenge for the practitioner is the choice of a suitable clustering tool for the dataset at hand; not every clustering tool performs equally well on each dataset [7]. Milligan wrote that "there is no single approach to clustering that can be regarded as appropriate for most situations" [83] and regards the choice of an appropriate clustering tool as a fundamental clustering problem. In order to make an informed decision, the researcher is required to have a solid knowledge in clustering algorithms in order to know (a) which approaches exist, (b) their limitations, and (c) how they are applied most efficiently. Ignorance of these facts may lead researchers to make sub-optimal choices [83]. The problem worsens as there is no repository where all available tools can be found. Furthermore, there is a lack of readily-available, unbiased information concerning the performance of the different tools on a sufficient number of datasets [83]. New clustering approaches tend to be tested solely against a handful of other algorithms. The impartiality of the datasets selected for this purpose is not guaranteed. Thus, even for experts, it is very hard to identify the best-suited

clustering tool for a specific problem [83, 47] and "in many applications it might be reasonable to apply a number of clustering methods" [47].

We intend to assist researchers in conducting a cluster analysis with our clustering framework ClustEval (refer to Chapter 6 on page 107) which automatizes most of the common processes in a cluster analysis. Refer to Section 1.2 on the facing page where we clarify the objectives of this thesis.

Challenges when Utilizing Current Clustering Approaches

- The plethora of clustering tools is confusing and selecting the most feasible tool for a task is challenging.
- There is no standard pipeline for conducting high quality clustering analyses.

Once the researcher has identified a feasible clustering tool, a meaningful cluster analysis still makes high demands on the researcher. Almost all clustering approaches require at least one parameter to be set specifying the behavior of the algorithm (compare to Section 2.4 on page 39). Generally, this parameter or parameter set is used to tune the tool to obtain optimal results, e.g., controlling the average size of or the number of clusters in the result. Finding such a parameter is a challenging task, as there is no generally accepted definition of a good clustering since the quality of a clustering is highly problem specific [127, 83]. Thus, proposing a generally accepted method for obtaining suitable parameters is impossible. Especially if no gold standard for a similar problem instance is available for training, the correct application of a clustering tool poses a challenge in practice. Furthermore, the bare result of a clustering alone does not necessarily lead to new insights. In order to derive new knowledge, these results must be integrated into the relevant biological context using further data from different sources. Again, there is no general approach for the meaningful handling of clustering results.

In the scope of this thesis, we will concentrate on the application of clustering for protein-homology detection. We will exemplarily demonstrate the application of TransClust in two studies and develop a method for reliably discovering a parameter set for protein homology detection. Refer to Section 1.2 on the facing page where we clarify the objectives of this thesis.

Challenges of Deriving Insights from a Cluster Analysis

- There are no generally accepted guidelines for setting the parameter(s) of a clustering tool.
- The clustering result is rarely the final analysis step in a study; frequently, it needs to be integrated with other methods and datasets.

1.2. Objectives of This Work

In the motivation we have highlighted three challenges of current-day cluster-analyses: (1) the abundance of biological data, (2) the plethora of different clustering tools and their utilization, and (3) the integration of clustering results in biological frameworks. In this section we want to summarize the objectives of the thesis and outline how we account for those challenges .

Missing Values The abundance of biological data and the resulting size of the datasets reaches the limits of current computational power. Especially the calculation of pairwise similarities consumes the biggest part of computational time spent during a cluster analysis. In order to cluster large datasets more efficiently, we exploit the fact that most similarities may be redundant and thus play only a minor role in the overall clustering result; thus, only parts of the similarity matrix are required for clustering. We test this assumption in two steps:

1. We enable TransClust to cope with *missing values* (TransClustMV). That means TransClust does not require a full pairwise proximity matrix anymore but can cluster a dataset when only part of the proximity matrix is available. This has several consequences:
   a) The researcher can intentionally omit the calculation of similarities in order to save runtime for the calculation and space for the storage of the proximity matrix.
   b) TransClust can also cluster similarity matrices which are unintentionally incomplete, e.g., since some similarities were never measured in a wet-lab experiment.
2. We develop an approach called ActiveTransClust in which the clustering tool does not solely cope with a similarity file with missing values, but actively chooses which similarity values must be calculated in an iterative process.

Summary of Objective "Missing Values"

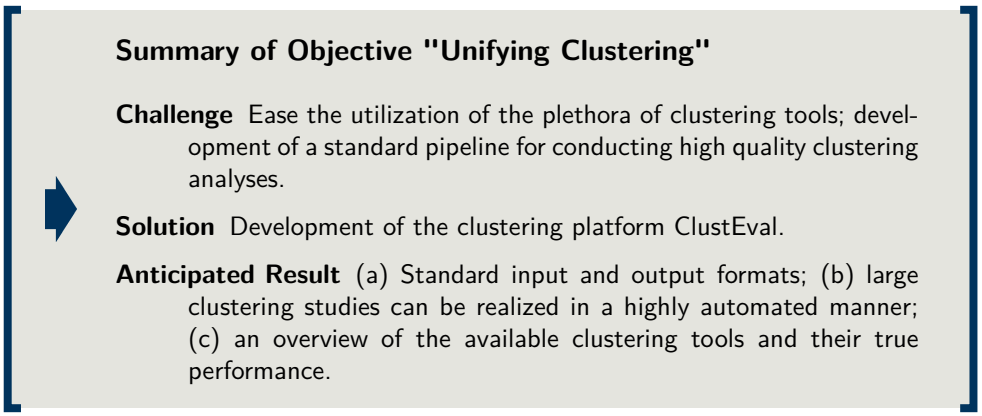
Challenge Enable the utilization of large similarity files and the usage of complicated similarity functions even for large datasets.

Solution Exploitation of missing values in order to reduce the amount of required similarity calculations.

Anticipated Result A clustering tool which produces high-quality results while only using a small fraction of the similarities.

Unifying Clustering As described in the motivation, we identified that the growing number of different clustering tools and data standards make the execution of a cluster analysis increasingly difficult. Here, we aim to ease conducting cluster analyses, especially comparative ones, using several tools and datasets. Furthermore, we want to give an overview of the wide variety of algorithms and data standards. We propose a standardized format for input and output files. We develop an entire clustering framework (called ClustEval) automating most of the typical steps of a cluster analysis. This high degree of automation will not only reduce the amount of time required for conducting a clustering study, but also increase the comparability of the tools' performances as all tools are tested following an identical protocol. Especially methods for the automated detection of the best density parameter will render the results created by this framework bias-free and reproducible as no tool is favored over another. We also present results of clustering runs of prominent datasets on a website, enabling non-experts to make an informed decision as to which clustering tool to use for which kind of dataset. The desired features of our clustering framework ClustEval can be summarized as follows:

1. ClustEval provides an overview of commonly used clustering tools in bioinformatics (refer to Section 2.4 on page 39) together with their performance on several biological datasets.
2. ClustEval automates the application of several clustering tools on a dataset.
3. ClustEval also automates the identification of a parameter set maximizing a given cluster quality measure (refer to Section 2.3 on page 32).



Clustering in Practice As a third goal, we demonstrate the power of clustering algorithms in real-world biological applications. Here, the practitioner faces problem-specific challenges which render it impossible to provide general guidelines. In order to ease the widely used and challenging task of clustering in biological applications, namely, protein homology detection [7, 29], we present

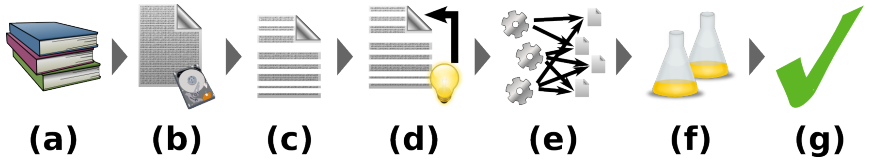


Figure 1.2.: Structure of this thesis. From left to right: We start with **(a)** the related work and background, then **(b)** enable the handling of large similarity files by utilizing background storage. Afterwards, we **(c)** introduce missing values, followed by **(d)** introducing active clustering. In the next part, we **(e)** introduce the ClustEval framework which is followed by **(f)** two studies with biological datasets. We finish this thesis with **(g)** a discussion, conclusion and an outlook.

two studies. One demonstrates the integration of a clustering into an entire analysis pipeline aiming to transfer gene regulatory networks of a model organism to several target organisms. In another study, we develop a reliable and stable method for detecting a meaningful threshold using only intrinsic information from the dataset. We will use this threshold in order to detect homologous proteins by means of clustering selected actinobacteria in order to unravel pathogenicity specific genes which may help to distinguish harmless bacteria from pathogens.

Summary of Objective "Clustering in Practice"

Challenge Integration of clustering results into a biological analysis pipeline; development of a guideline to detect a meaningful threshold for protein homology detection without a gold standard.

Solution Using intrinsic information of the dataset to derive a suitable threshold; integration of the clustering tool in an evolutionary data analysis pipeline.

Anticipated Result (a) a robust method for detecting a threshold without a gold standard; (b) evolutionarily conserved networks.

1.3. Structure of This Work

Chapter 1: In this chapter, the motivation for this thesis is presented and the objectives are defined.

Chapter 2: This chapter provides the reader with all basic definitions and the general design of a cluster analysis. In the remainder, the most common proximity measures, clustering approaches, and cluster-quality measures are presented.

Chapter 3: In this chapter, an efficient method for decomposing large similarity files into connected components is presented. This method circumvents the shortage

of main memory by utilizing background storage while still being computationally effective. At the end, runtime analyses are presented.

Chapter 4: This chapter describes the decomposition of cost-matrices while strategically omitting the calculation of many similarities. The cluster quality is compared to a clustering with full information using different datasets.

Chapter 5: Active clustering is introduced in this chapter. First, problems with the method presented in the previous chapter are discussed followed by an analysis and strategy of how to evaluate the importance of missing values. Again, the results are compared against a clustering based on full information.

Chapter 6: In this chapter, the ClustEval framework is introduced. After describing the concept, the chapter explains how to create artificial datasets and efficient ways for probing for an optimal threshold. The results are summarized by giving three example usecases for the framework and comparing them to a manual study design.

Chapter 7: Here, two real-world applications are presented. (1) EHECRegNet, which demonstrates the integration of clustering results into an entire analysis pipeline transferring evolutionarily conserved gene regulation from a model organism to several target organisms. (2) We develop a reliable method for calculating a meaningful density parameter for protein homology detection using only intrinsic information of the dataset.

Chapters 8 & 9: Discussion and conclusion of the presented methods and studies, their powers and limitations. In comparison to the previous chapters which also contain a small discussion and conclusion, we focus in these chapters on the overall context of the entire thesis.

Chapter 10: In the last chapter, we give an outlook on possible future developments and suggest follow-up studies to overcome some of the limitations outlined in the discussion chapter.

Figure 1.2 depicts the structure of the work and is displayed at the beginning of each chapter to ease the reader's orientation. Furthermore, summaries are displayed in a box like the following (as already seen in the motivation):

The small arrow on the left side is changed to a check (✔) when findings of a chapter are summarized. Outlooks are indicated with a different box:

1.4. Publications

In the framework of this thesis several scientific papers emerged. The thesis itself is highly based on the following publications:

Richard Röttger, Prabhav Kalaghatgi, Peng Sun, Siomar de Castro Soares, Vasco Azevedo, Tobias Wittkop, Jan Baumbach. **Density parameter estimation for finding clusters of homologous proteins - tracing actinobacterial pathogenicity lifestyles.** *Bioinformatics* (2013), 29 (2), 215-222, DOI:10.1093/bioinformatics/ bts653
Section 7.2 is based on this publication.

Richard Röttger, Jan Baumbach. **How Little Do We Actually Know? - On the Size of Gene Regulatory Networks.** Shortened form for the highlight track of the *German Conference on Bioinformatics 2012* (original publication appeared in the IEEE/ACM transactions on computational biology and bioinformatics).
Findings of this publication are used in Subsection 7.2.3.

Richard Röttger, Christoph Kreutzer, Thuy Duong Vu, Tobias Wittkop, Jan Baumbach. **Online Transitivity Clustering of Biological Data with Missing Values.** *German Conference on Bioinformatics 2012*, 57-68, ISBN: 978-3-939897-44-6, DOI: 10.4230/OASIcs.GCB.2012.57
Chapter 4 is based on this publication.

Richard Röttger, Ulrich Rückert, Jan Taubert, Jan Baumbach. **How Little Do We Actually Know? - On the Size of Gene Regulatory Networks.** *IEEE/ACM transactions on computational biology and bioinformatics* (2012), 9(5), 1293-1300, DOI: 10.1109/TCBB.2012.71.
Findings of this publication are used in Subsection 7.2.3.

Richard Röttger[4], Josch Pauling[4], Andreas Neuner, Heladia Salgado, Julio Collado-Vides, Prabhav Kalaghatgi, Vasco Azevedo, Andreas Tauch, Alfred Pühler, Jan Baumbach. **On the trail of EHEC/EAEC - Unraveling the gene regulatory networks of human pathogenic *Escherichia coli* bacteria.** *Integrative Biology.* 2012 Jul;4(7):728-33. doi: 10.1039/c2ib00132b.
Section 7.1 is based on this publication.

Josch Pauling, Richard Röttger, Andreas Tauch, Vasco Azevedo, Jan Baumbach. **CoryneRegNet 6.0 - Updated database content, new analysis methods and novel features focusing on community demands.** *Nucleic Acids Res.* (2012) Jan;40(1):D610-4.
Findings of this publication are used in Section 7.1.

[4]Joint first authorship of Richard Röttger and Josch Pauling.

Tobias Wittkop, Sven Rahmann, Richard Röttger, Sebastian Böcker, Jan Baumbach. **Extension and robustness of Transitivity Clustering for protein-protein interaction network analysis.** *Int. Math.* (2011), 7:4, 255-273.
Findings of this publication are used in Section 2.4 and in Chapter 3, 4, and 5.

The following four publications are either submitted or in preparation:

Richard Röttger, Christoph Kreutzer, Sebastian Böcker, Sven Rahmann, Tobias Wittkop, Jan Baumbach. **Fuzzy Transitivity Clustering for unraveling complexes in protein-protein interaction networks.** (submitted)
Findings of this publication are used in Section 2.4 and Chapter 6.

Peng Sun, Nora K. Speicher, Richard Röttger, Jiong Guo, Jan Baumbach. **Bi-Force - Large-scale bicluster editing and its application to gene expression data biclustering.** (accepted, *Nucleic Acids Res.*)
Findings of this publication are used in Chapter 6.

Richard Röttger, Alexander Junge, Jan Baumbach. **Active Transitivity Clustering.** (working title, in preparation)
Chapter 5 is based on this publication.

Christian Wiwie, Jan Baumbach, Richard Röttger. **Standardization and Evaluation of Popular Bioinformatics Clustering Tools - An Integrated Online Framework.** (in preparation)
Chapter 6 is based on this publication.

2. Background and Related Work

2.1. Overview

2.1.1. Basic Definitions

As this thesis focuses on transitivity clustering, a graph-based clustering method, the basic definitions of a graph must be given.

Definition 2.1.1. Undirected Simple Graph A graph G is a pair of sets $G = (V, E)$ with $E \subseteq [V]^2$. Normally, the set V is called *vertices* or *nodes* whereas the set E compromises the *edges*. Here, $[V]^2$ denotes all two-element subsets of V, which means the graph neither contains self-loops nor multiple edges between a pair of nodes. We will use $uv \in E$ as shorthand for $\{u, v\} \in E$ to denote edges.

Definition 2.1.2. Directed Graph A directed graph $G = (V, E)$ again consists of a set of vertices V but the set of edges is now defined as $E \subseteq V \times V$. Let $(u, v) \in E$ be an edge of the graph, by convention we define u as the start of the edge and v as the end. We also use uv as a shorthand for directed edges. With this definition, self-loops are allowed.

Definition 2.1.3. Subgraph A graph $G' = (V', E')$ is called a subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. In short, we write $G' \subseteq G$.

Definition 2.1.4. Induced Subgraph A subgraph G' of G is called an induced subgraph, iff G' contains all edges $uv \in E$ with $u, v \in V'$.

Definition 2.1.5. Path A path $P = (V, E)$ is a non-empty graph of the form

$$V = \{v_0, \ldots, v_k\} \qquad E = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$$

with all v_i being distinct. The *length* of a path is defined as the number of edges in this path. The notation of a path can be shortened as $P = v_0v_1 \ldots v_k$. Commonly, a path P is a subgraph of a graph G.

Definition 2.1.6. Connected Component A connected component $G' = (V', E')$ is defined as an induced subgraph of an undirected simple graph $G = (V, E)$ such

that (1) there is a path between all pairs of nodes in V' and (2) there exists no edge uv with $u \in V'$ and $v \in V \backslash V'$. In other words, no vertex can be added to V' without violating requirement (1). For an directed graph we call this defines a *strongly connected component.*

Definition 2.1.7. Transitivity We call a graph $G = (V, E)$ transitive if for every triplet $\{u, v, w\} \in \binom{V}{3}$

$$uv \in E \wedge vw \in E \Rightarrow uw \in E$$

holds true.

2.1.2. Definition of a Clustering

To begin, we need to establish a definition of a clustering. As already mentioned, a cluster analysis is performed whenever hidden traits in a dataset need to be exposed. Nevertheless, due to the many different application areas of clusterings, it is almost impossible to find a definition fulfilling all demands to a clustering for all different scientific areas [127, 83]. As Xu *et al.* [127] put it, "there exists no universally agreed-upon and precise definition of the term cluster, partially due to the inherent subjectivity of clustering, which precludes an absolute judgment as to the relative efficacy of all clustering techniques". In this work however, we will give a definition of clustering based upon a proximity measure, i.e., a similarity measure. We are looking for a partition of a given set of objects, such that the objects within a cluster are more similar to each other (*intra-cluster similarity*) than objects of different clusters (*inter-cluster similarity*) to each other on average.

In the focus of this work, we distinguish between three different kinds of clustering. We assume to have a set of N objects $V = \{v_1, \ldots, v_N\}$

Definition 2.1.8. Partitional Clustering The *(hard) partitional clustering* (also called *disjoint* or *crisp clustering*) assigns each object $v_i \in V$ to exactly one cluster C_j. A clustering $C = \{C_1, \ldots, C_K\}$ with $K < N$ is a subset of the power set $\mathcal{P}(V)$ for which holds true:

1. $C_i \neq \emptyset \qquad \forall i = 1, \ldots, K$
2. $\bigcup_{i=1}^{K} C_i = V$
3. $C_i \cap C_j = \emptyset \qquad \forall i, j = 1, \ldots, K \,\text{and}\, i \neq j$

In the remainder of the manuscript, we refer to this type just as *clustering.*

Definition 2.1.9. Fuzzy Clustering The *fuzzy clustering* follows a similar intuition as the partitional clustering. Again, we consider a clustering $C = \{C_1, \ldots, C_K\}$ as a subset of of the power set $\mathcal{P}(V)$. In contrast to hard partitional clustering, with fuzzy clustering, each object v_i can belong to several clusters to a certain degree. Therefore, we define a membership of v_i to cluster C_j as $u_{i,j} = [0, 1]$. For a valid clustering, the following properties hold true:

1. $\sum_{i=1}^{K} u_{i,j} = 1 \qquad \forall j$
2. $0 < \sum_{j=1}^{N} u_{i,j} < N \qquad \forall i$

The first condition assures that all objects are contained completely in the clustering, the second condition forbids the existence of empty clusters analogously to the points 1 and 2 to the definition of the hard partitional clustering.

Definition 2.1.10. Overlapping Clustering The *overlapping clustering* is similar to the fuzzy clustering. In contrast to fuzzy clustering, the elements of an overlapping clustering $C = \{C_1, \ldots, C_K\}$ do not have a degree of their membership and is defined as follows:

1. $C_i \neq \emptyset \qquad \forall i = 1, \ldots, K$
2. $\bigcup_{i=1}^{K} C_i = V$

Definition 2.1.11. Hierarchical Clustering In contrast to both aforementioned clusterings, hierarchical clustering does not simply partition a set into a unique clustering but rather builds a nested structural partition. Again, we consider a clustering $C = \{C_1, \ldots, C_K\}$ as a subset of the power set $\mathcal{P}(V)$ such that $\bigcup_{i=1}^{K} C_i = V$ and $C_i \neq \emptyset$ for $i = 1, \ldots, K$. Furthermore, for each pair C_i, C_j of clusters with $i, j = 1, \ldots, K$ and $i \neq j$, exactly one of the following conditions holds:

1. $C_i \cap C_j = \emptyset$
2. $C_i \subset C_j$
3. $C_j \subset C_i$

2.1.3. Cluster Analysis

Despite the large variety of problems tackled with a cluster analysis, all applications follow a certain structure, which is also depicted in Figure 2.1. They are based on the concepts presented in [57, 48, 59]:

1. *Data acquisition and preparation:* This step refers to the process of acquiring the raw-data and processing them to a ready-to-use similarity file as an input for a clustering tool. This process itself can be subdivided into the following steps:
   a) Data acquisition: all data involved in the cluster analysis is acquired. In the framework of the thesis, these are datasets generated in wet-lab experiments or artificial generated data.
   b) Pattern representation: In this step, the features relevant for the cluster analysis are selected and/or extracted [48, 57]. Feature selection describes the search and evaluation of a set of features of the dataset which are best

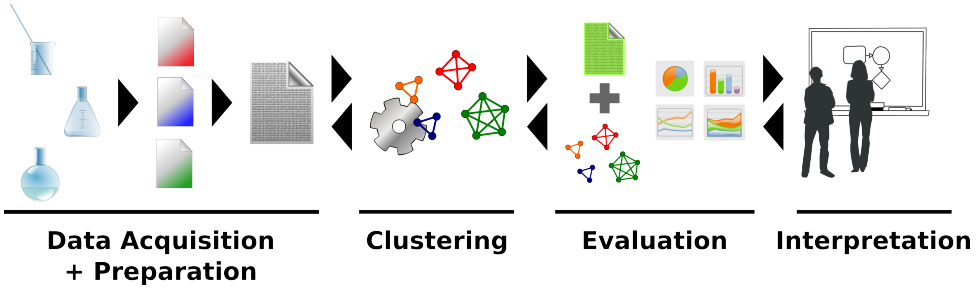


Figure 2.1.: Workflow of a common cluster analysis. All data required for the analysis, for example measurements of wet-lab studies, need to be homogenized and transfered to a single similarity file which forms the input for the clustering algorithm. The next step is the actual execution of the clustering, followed by the evaluation. The evaluation can be the comparison of reported clusters with a gold standard or some other measure for judging the result quality. After the best possible clustering is found, the actual result can be used and interpreted.

suited for the clustering. In contrast to feature selection, feature extraction involves transformations of the original features into new features that are more appropriate for the clustering (e.g., principal component analysis, dimension reduction, etc.) [48].

c) Similarity measure: Here, a suitable proximity measure is selected. Such a similarity function is highly problem dependent. In Section 2.2 on the next page, common proximity measures are described. Generally, "it is important to check that all selected features contribute equally to the computation of the proximity measure and that no features dominate others" [48].

2. *Clustering:* In the next step, the actual clustering is performed. Generally, the selection of the most suitable algorithm and its correct application requires experience in clustering from the scientist in order to achieve optimal results. Normally, this is done by defining “the clustering criterion, which can be expressed via a cost function or some other type of rules” [57] first, then selecting an algorithm most suited for optimizing that criterion (e.g., minimizing the average distance of objects to their cluster center for a fixed number of cluster). Once an algorithm is selected, finding optimal the parameters for the algorithm is a challenging task as well. Several prominent algorithms used in bioinformatics are introduced in Section 2.4 on page 39.

3. *Evaluation:* In this step, the cluster results are evaluated. Whenever a gold standard is available, *external quality* measures can be applied which assess the agreement of the reported clustering and the gold standard. In contrast, if there is no gold standard available, so called *internal* measures are utilized. These measures evaluate a clustering based on statistical properties of the result, e.g., the distribution of the intra- vs. inter-cluster similarity (refer to Section 2.3 on page 32 for more details on quality measures). The result of the evaluation

process is used to optimize the parameter settings of the applied algorithm by re-performing the clustering step. Note that in Section 7.2 on page 136, we introduce a novel internal measure specifically tailored for protein homology detection.

4. *Interpretation:* In the last part of a cluster study, the results are interpreted and a hypothesis can be checked or formulated. It is important that steps 3 and 4 are not interchanged, meaning that the clustering must not be optimized to support the researcher's hypothesis best, but should follow a preferably objective and independent (from the hypothesis) measure.

In order to receive good quality results, each step must be carried out as carefully as possible, as mistakes or biases are carried forward through the entire analysis. All steps should be carried out with a feedback loop (as depicted in Figure 2.1), thus results in one step may cause revised decisions in a previous step [59]. In order to support practitioners with such a tedious and error-prone cluster analysis, we developed an integrated clustering framework, ClustEval (refer to Chapter 6 on page 107), which assists the user through the aforementioned steps 1-3.

2.2. Similarity Functions & Preprocessing

Once the data is acquired, several steps of preprocessing need to be performed in order to convert the data at hand into a readable format for clustering algorithms. A common trait of all clustering approaches discussed in this thesis is the necessity of a proximity measure. Such a proximity measure quantifies how "close" or how "distant" two objects are to each other [48]. These measures are called *similarity* or *distance* measure respectively. Note that distance measures can be transformed into a similarity measure and vice versa. Depending on the data type (continuous or categorical), a multitude of different measures exist. In the framework of this thesis, we focus on continuous datasets. Furthermore, we will handle some special cases used in bioinformatics, like protein similarities based on their amino acid sequence. Generally, the raw data is available in a $N \times d$ *data matrix* representing N different objects described with d different attributes. The aim of the proximity function is the calculation of the proximity measures stored in a symmetrical $N \times N$ *similarity matrix.*

2.2.1. Metric Based Distances

Definition 2.2.1. Metric A function $m : V \times V \to \mathbb{R}$ on an arbitrary set V is called metric on V if for any $u, v, w \in V$ holds:

1. $m(u, v) \geq 0$
2. $m(u, v) = 0 \quad \Leftrightarrow \quad u = v$
3. $m(u, v) = m(v, u)$

4. $m(u, w) \leq m(u, v) + m(v, w)$

Refer to Gower *et al.* [56] for an extensive discussion about different metric dissimilarity measures. Nonetheless, the most commonly used distance measures belong to the family of the Minkowski distance.

Definition 2.2.2. Minkowski distance Let $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$ be two objects of the set V. The *Minkowski distance* is now defined for every $p \geq 1$ as

$$m_p(u, v) = \left(\sum_{k=1}^{d} |u_k - v_k|^p \right)^{\frac{1}{p}}$$

The best known representatives for this class of metrics is the *Euclidean distance* with $p = 2$,

$$m_2(u, v) = \left(\sum_{k=1}^{d} |u_k - v_k|^2 \right)^{\frac{1}{2}},$$

the *City-block* or *Manhattan distance* with $p = 1$,

$$m_1(u, v) = \sum_{k=1}^{d} |u_k - v_k| ,$$

and the *Sup distance* with $p = \infty$,

$$m_\infty(u, v) = max_{1 \leq k \leq d} |u_k - v_k| .$$

All metrics mentioned above also have disadvantages which should be considered when using such a metric. Every dimension is weighted equally which can potentially lead to a domination of one dimension over the others. As an example, consider data points in two-dimensional Euclidean space where the values of dimension 1 range between $[0, 1]$ and of dimension 2 between $[0, 100]$. Let $x = (0, 20)$ and $y = (1, 50)$. Using a metric, the distance between x and y is clearly dominated by the second dimension even though the difference of the first dimension is more significant in relative terms (as it spreads over the entire range of that dimension). To counter that, the data can be normalized, so that all dimensions have the same range. A feasible normalization is very problem-dependent and would exceed the scope of this thesis. Commonly used examples can be found in [47].

2.2.2. Correlation-based Distances

A very popular class of distance measures is presented by measures based on the correlation coefficient. In bioinformatics, the correlation coefficient is often used for analyzing gene expression studies [41]. This is mainly due to the property of the correlation that it is oblivious to the span of magnitudes of the observed values [47]. The correlation measures the similarity of the observations' relative profiles [47] which

matches the intuition of genes being co-expressed [41]. For example if we look at two genes u and v which show the following "expression levels" in three patients of $v = (1, 2, 1)$ and $u = (3, 6, 3)$, we want them to be labeled as co-expressed, as their expression pattern follows the same relative profile. In one of the aforementioned metrics, these genes would be rather dissimilar. Therefore, the correlation coefficient should only be used in cases where all the data was measured on the same scale and the actual value is not of particular interest [47].

Definition 2.2.3. Pearson Correlation Let $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$ be two objects of the set V. Furthermore, let $\overline{w} = \frac{1}{d}\sum_{k=1}^{d} w_k$ the average of $w \in V$. The Pearson correlation $\delta(u, v)$ is now defined as

$$\begin{aligned} \delta(u,v) &= \frac{(1-\phi(u,v))}{2} \qquad \text{with} \\ \phi(u,v) &= \frac{\sum_{k=1}^{d}(u_k-\overline{u})(v_k-\overline{v})}{\left(\sum_{k=1}^{d}(u_k-\overline{u})^2 \cdot \sum_{k=1}^{d}(v_k-\overline{v})^2\right)^{\frac{1}{2}}} \end{aligned}$$

Definition 2.2.4. Spearman Correlation The Spearman correlation is the Pearson correlation between the ranked variables $rg(u)$ and $rg(v)$ with

$$\begin{aligned} rg(u) &= (rg(u_1), \ldots, rg(u_d)) \\ rg(v) &= (rg(v_1), \ldots, rg(v_d)). \end{aligned}$$

That means, every real value of a point $u \in V$ is ranked according to a given ranking function rg. Generally, any ranking function rg may be used. The final Spearman correlation $\rho(u, v)$ is defined as

$$\rho(u, v) = \delta(rg(u), rg(v))$$

with δ being the Pearson correlation.

The Spearman correlation is less sensitive to large outliers [85]. Furthermore, the Pearson correlation yields perfect results when the two variables are related linearly, the Spearman correlation only requires a monotonic relation for a perfect correlation. Thus, when testing for a linear correlation, the Pearson correlation should be favored over the Spearman correlation.

2.2.3. BLAST

Besides the metric and correlation based distances, there exist specialized similarity functions for certain data types. Protein sequences are an example for such a data type. Here, we introduce a common method for calculating similarities between protein amino acid sequences based on the sequence alignment of the Basic Local Alignment Search Tool (BLAST) [5, 6, 25].

BLAST is used to create local alignments of the input sequences (called *query sequences*) of proteins with an already existing database of sequences (called *subject*

sequences). For a cluster study, a so-called *all-vs.-all* search is performed, which means that the database contains the same sequences which are queried afterwards.

BLAST divides the query sequence into small "words", typically of length three [6]. The database is searched for these words and whenever a "good" hit is found, the hit is expanded to a local alignment. Such a local alignment is then reported as a *High Scoring Pair* (HSP). The most important result for a cluster analysis is the reported E-value of such a HSP. The E-value reports the number of expected hits which would be found by mere chance in a random database of the same size. In other words, the lower the E-value, the more significant the reported hit. In order to transfer the E-value into a similarity value, the negative log-likelihood is calculated. For two proteins u and v, the similarity s is $s(u,v) = -\log_{10}(\text{E-value})$.

One characteristic of BLAST is that it can report several HSPs for every pair of proteins u and v in both directions (direction means if u is aligned to v or v is aligned to u). Let $(u \leftarrow v)_k$ and $(u \rightarrow v)_l$ with $k = 1, \ldots, K$ and $l = 1, \ldots, L$ be the HSPs of u and v in the indicated direction. Following the suggestions of Wittkop [120], there are three different means of combining these HSPs to one similarity value.

Definition 2.2.5. Best Hit (BeH) This method selects the best HSP in each direction (the HSP with the lowest E-value) and in order to "play save" (reduce the number of false positives), chooses the worst of these two HSPs (the one with the highest E-value):

$$s(u,v) = -\log\left(\max\left\{\min_{k=1,\ldots,K} \text{E-value}((u \leftarrow v)_k), \min_{l=1,\ldots,L} \text{E-value}((u \rightarrow v)_l)\right\}\right).$$

Definition 2.2.6. Sum of Hits (SoH) This method combines all HSPs in each direction and afterwards chooses the worst direction for the final similarity:

$$s(u,v) = -\log\left(\max\left\{\prod_{k=1}^{K} \text{E-value}((u \leftarrow v)_k), \prod_{l=1}^{L} \text{E-value}((u \rightarrow v)_l)\right\}\right).$$

Definition 2.2.7. Coverage (Cov) The last approach additionally accounts for the length of the two proteins in question. For example the perfect match of a short protein to a very long one would result in a good HSP, but the biological homology of both proteins is quite unlikely (given the large difference length alone). In order to judge the coverage, we define an indicator function:

$$\mathbb{I}_{uv}(p) = \begin{cases} 1 & \text{if in } u \text{ the position } p \text{ is covered by any HSP } (u \leftarrow v)_k \text{ or } (u \rightarrow v)_l \\ 0 & \text{otherwise.} \end{cases}$$

The coverage is again the worst-case of both directions:

$$\text{coverage}(uv) = \min\left(\frac{1}{|u|}\sum_{p=1}^{|u|} \mathbb{I}_{uv}(p), \frac{1}{|v|}\sum_{p=1}^{|v|} \mathbb{I}_{uv}(p)\right).$$

This function does not yet account for the quality of the coverage. Letting s' be either the BeH or SoH similarity function, the Cov similarity is defined as

$$s(u, v) = s'(u, v) + \lambda \cdot \text{coverage}(u, v)$$

with λ being a parameter defining the importance of the coverage. As the coverage is a measure between $[0, 1]$, λ has to be chosen large enough to compete with the similarities calculated by BeH or SoH which approximately range between $[0, 200]$, depending on the implementation. In several studies a λ between 15 and 25 has proven useful for protein homology detection [121, 120].

2.2.4. Converting Distances into Similarities

In this section we introduced both distance measures and similarity measures and treated them equally by just naming them proximity measures as both describe the relationship between two objects. On the other hand, each clustering tool requires a specific type of proximity measure. TransClust requires a similarity measure, for instance.

As already mentioned above, similarities and distances can be converted into each other. Here we will describe some general concepts to convert a distance into a similarity; in practice, however, a meaningful conversion depends on the clustering algorithm (e.g., some require similarities between $[0, 1]$) and the distance used. If the given distance is limited, e.g., within $[0, 1]$, the conversion can be performed rather easily. Let $d(u, v)$ denote the distance between the two objects $u, v \in V$; the application of one of the following formulas results in a similarity $s(u, v)$:

$$\begin{aligned} s(u, v) &= 1 - d(u, v) \\ s(u, v) &= \sqrt{1 - d(u, v)} \\ s(u, v) &= -\log(d(u, v)) \\ s(u, v) &= \frac{1}{d(u, v)} - 1 \quad \forall d(u, v) \neq 0 \end{aligned}$$

Which formula to apply depends highly on the nature of the distance function. For example, BLAST E-values get converted by applying $-\log(\text{E-value})$.

If the range of the values of a distance function is not bound to an upper limit, the idea of the above conversions needs to be generalized. That can be achieved by applying a monotonically decreasing function [60]. For example, the euclidean distance $m_2(u, v)$ is normally converted to a similarity by

$$s(u, v) = \frac{1}{1 + m_2(u, v)}$$

or as suggested for example in the paper of Strahl *et al.* [109] with

$$s(u, v) = e^{-m_2(u,v)^2}.$$

These are some examples of how to convert distances into similarities and why we treat them in the entire thesis as equivalent in practice. Again, the specific best conversion highly depends on the problem at hand and the distribution of the distances in the given dataset.

2.2.5. Missing Values

In the course of this work, we will concentrate on missing values and their influence on clustering algorithms. We can distinguish between two different types of missing values which must not be confused.

Definition 2.2.8. Implicitly Missing Values The first type of missing values is what we call *implicitly missing values*. Depending on the type of data used it can happen that some values for the similarity matrix are missing as they fall below a certain detection limit. For example, when using BLAST, the user can specify a so-called E-value cut-off. Here, BLAST discards the calculation of a HSP whenever a certain similarity cannot be reached anymore; thus, the result file is missing this pair. In fact, nevertheless, these missing values do carry information, namely that the actual value is very low. Most clustering algorithms treat these missing values by setting them to the lowest possible similarity (depending on the function) or a certain user defined value. In practice this is usually done to speed up computing the similarity matrix.

Definition 2.2.9. Explicitly Missing Values In contrast to the implicitly missing values, we do not have any meaningful assumption about the actual value of explicitly missing values. These missing values arise by omitting the calculation of the similarity in the first place, or by the design of the dataset. For example, when the dataset reflects some wet-lab experiment, it is quite usual that not all pairs were actually measured. In that case, just because it was never measured, we cannot assume the similarity to be low or any other predefined value. In fact, we may not draw any assumption. That poses a problem for clustering algorithms because they cannot be filled up with some fall-back value; the algorithm must find a way to deal with them without distorting the entire clustering result.

Here, we will focus mainly on explicitly missing values as we will exploit them in order to reduce the computational effort required to calculate the similarity file.

2.3. Quality Measures

A crucial part of a cluster analysis is the evaluation of the reported clusters in order to train or optimize the algorithm's parameters. Two main types of quality measures can be distinguished [59, 73]:

Internal measures Internal quality measures judge the clustering C by certain intrinsic statistical properties of the clustering itself.

External measures External measures compare the clustering result C with an available gold standard K.

The latter method is more straightforward but poses the problem of a possible overtraining of the parameters and is, of course, highly dependent upon the quality of the gold standard. At the end of this section, we provide on overview of the discussed quality measures and provide guidelines when to use a certain quality measure.

2.3.1. Internal Quality Measures

One often used internal quality measure was introduced in 1974 by J.C. Dunn [38]. The intuition is that the clusters of a good clustering are compact and well-separated with regards to the distance measure d. Therefore, a cluster diameter for a cluster C_i is defined as

$$\operatorname{diam}(C_i) = \max_{u,v \in C_i} d(u,v)$$

and an inter-cluster distance between two clusters C_i, C_j as

$$d(C_i, C_j) = \min_{u \in C_i; v \in C_j} d(u,v).$$

Definition 2.3.1. Dunn index The *Dunn index* is defined as

$$D(C) = \min_{C_i \in C} \left\{ \min_{\substack{C_j \in C \\ C_i \neq C_j}} \left\{ \frac{d(C_i, C_j)}{\max_{C_p \in C} \operatorname{diam}(C_p)} \right\} \right\}.$$

A clustering C is said to consists of *compact separated* clusters relative to d iff $D(C) > 1$.

The Dunn index is overly sensitive to noise as just one outlier can dramatically change the diameter of an otherwise very compact cluster or the distance between two clusters [21]. Thus, several other indexes based on the Dunn index have been proposed using different methods for defining the diameter and inter-cluster distance, e.g., using the cluster centers for the distance.

The central idea of the Davies-Bouldin index is to assess the average similarity of each cluster to its most similar cluster [32]. For that index, we consider a d-dimensional feature space and the usage of a Minkowski distance as the proximity measure. In their original work, Davies *et al.* [32] do not restrict themselves to a certain metric; however, in the scope of this work, we concentrate on the case of

using a Minkowski distance rather than the generalized approach. Let $\overline{C}_i$ denote the centroid of cluster C_i. We define a scatter measure σ_i of cluster C_i as

$$\sigma_i = \sqrt[p]{\frac{1}{|C_i|} \sum_{c \in C_i} |c - \overline{C}_i|^p}$$

using the p as in the Minkowski distance. The distance between two clusters C_i, C_j is defined as the distance $m(\overline{C}_i, \overline{C}_j)$ of the corresponding centroids according to the metric used. The similarity measure R_{ij} between two clusters C_i, C_j is usually defined as

$$R(C_i, C_j) = \frac{\sigma_i + \sigma_j}{m(\overline{C}_i, \overline{C}_j)}.$$

Definition 2.3.2. Davies-Bouldin index The *Davies-Bouldin index* (DB) is defined as

$$\begin{aligned} \mathrm{DB}(C) &= \frac{1}{|C|} \sum_{C_i \in C} R(C_i) \quad \text{with} \\ R(C_i) &= \max_{\substack{C_j \in C \\ C_i \neq C_j}} R(C_i, C_j) \end{aligned}$$

A lower DB index indicates a better cluster quality. The DB index is often used to determine the number of clusters k when using K-means (the k resulting in the lowest DB index) [32, 127].

The next measure, the silhouette value, is based on a similar approach but in contrast to the aforementioned measures, factors in all pairwise dissimilarities. It was first introduced by Rousseeuw in 1986 [103]. For every object $u \in V$, $a(u)$ denotes the average dissimilarity of the object to its own cluster C_i. For the Euclidean distance, $a(u)$ is the average distance of u to all other objects of the cluster C_i. In contrast, for every cluster C_j with $j \neq i$, $d = (u, C_j)$ denotes the average dissimilarity to the cluster C_j. Further, let $b(u)$ be defined as

$$b(u) = \min_{\substack{C_j \in C \\ C_i \neq C_j}} d(u, C_j).$$

Definition 2.3.3. Silhouette value Following the definitions of $a(u)$ and $b(u)$ from above with $u \in V$, the silhouette value $s(u)$ is now defined to reward objects which are on average closer to their own cluster than to the closest neighboring cluster:

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

For the entire clustering C, the silhouette value is defined as the average silhouette of all N objects of the object set V:

$$s(C) = \frac{1}{N} \sum_{u \in V} s(u).$$

It is apparent that the silhouette values vary between $[-1, 1]$. For singletons, i.e., all clusters C_i with $|C_i| = 1$, the silhouette value is defined as 0. This measure is more robust than the other measures as a single outlier does not determine the entire measure for a cluster. It is also very common to plot the silhouette values of all objects sorted by the size of the cluster and their value.

2.3.2. External Quality Measures

Generally, external quality measures compare a clustering $C = \{C_1, \ldots, C_n\}$ received as a result of a cluster analysis with an existing gold standard or reference clustering $K = \{K_1, \ldots, K_m\}$. Most external measures are base on the concept of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). We distinguish between two different approaches to define these values:

Pairwise This approach compares the relationship between a pair of objects $u, v \in V$ in the clustering C and the reference clustering K. Each pair is counted as TP, TN, FP, or FN according to the following rules:

- **TP** Objects $u, v \in K_i$ for some i and $u, v \in C_j$ for some j.
- **TN** Object $u \in K_i$ and $v \in K_{i'}$ with $i \neq i'$ and $u \in C_j$ and $v \in C_{j'}$ with $i \neq j'$.
- **FP** Object $u \in K_i$ and $v \in K_{i'}$ with $i \neq i'$ but $u, v \in C_j$ for some j.
- **FN** Objects $u, v \in K_i$ for some i but $u \in C_j$ and $v \in C_{j'}$ with $i \neq j'$.

Mapping In this approach, each cluster K_i from the gold standard gets mapped to a cluster $c(K_i) \in C$. For each cluster $K_i \in K$ we can define an object $v \in V$ as TP, TN, FP, or FN according to the following conditions:

- **TP** $v \in K_i$ and $v \in c(K_i)$.
- **TN** $v \notin K_i$ and $v \notin c(K_i)$.
- **FP** $v \notin K_i$ but $v \in c(K_i)$.
- **FN** $v \in K_i$ but $v \notin c(K_i)$.In 1971, William M. Rand introduced an approach to comparing clusterings by comparing object pairs of the clusterings [99].

Definition 2.3.4. Rand index Using the *pairwise* definition of TP, TN, FP, and FN, the *Rand index* is defined as

$$R(K, C) = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\binom{N}{2}}$$

which corresponds to the number of agreeing pairs divided by the total number of pairs.

The Rand index varies between 0 and 1, with 0 meaning no agreement at all and 1 corresponding to two identical clusterings. There also exists an *adjusted Rand index* which factors in the expected agreement between two clusterings when the objects get assigned randomly to clusters.

The Rand index can be regarded as an extension of the Jaccard's coefficient, originally published in 1908. In contrast to the Rand index, the Jaccard index does not consider "true negatives", i.e., pairs of objects which belong in both clusterings to different clusters.

Definition 2.3.5. Jaccard index Using the *pairwise* definition of TP, TN, FP, and FN, the *Jaccard index* is defined as

$$J(K, C) = \frac{TP}{TP + FN + FN}.$$

The next measures follow a different approach as they do not utilize object pairs but rather compare entire clusters against each other. The most intuitive way of comparing two clusters is assessing their overlap in terms of common elements.

Definition 2.3.6. Overlap Score The *overlap score* is defined as

$$\omega(K_i, C_j) = \frac{|K_i \cap C_j|^2}{|K_i||C_j|}.$$

An overlap score of 0.25 or greater corresponds to a 50% agreement, given K_i and C_j are of the same size. The *fraction of matched clusters* is the number of clusters exceeding an overlap score of 0.25, divided by the total number of clusters in C.

All following measures are based on the mapping approach. The mapping of clusters $K_i \in K$ to clusters $C_i \in C$ is normally achieved by maximizing a given function. In order to simplify the notation, we first define the confusion matrix.

Definition 2.3.7. Confusion matrix The matrix $T = (t_{ij}) \in \mathbb{N}^{m \times n}$,

$$t_{ij} = |\{K_i \cap C_j\}|, \qquad 1 \leq i \leq m, \quad 1 \leq j \leq n$$

defines the *confusion matrix*, i.e., the overlap of all reference clusters with all clusters received from the algorithm.

Furthermore, we define $|T|$ as the sum of all entries in the confusion matrix, $|T_{\bullet j}|$ as the sum of all entries in jth column, and $|T_{i\bullet}|$ as the sum of all entries in the ith row.

Definition 2.3.8. Positive predictive value The *positive predictive value* (PPV) between a pair of clusters K_i, C_j is defined as the ratio between correct assignments (TP) to a reference cluster and all assignments to that reference cluster ($TP + FP$),

$$\mathrm{PPV}(K_i, C_j) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{t_{ij}}{|T_{\bullet j}|}.$$

Furthermore, for each cluster C_j the cluster-wise PPV is defined as

$$\mathrm{PPV}(C_j) = \max_i \mathrm{PPV}(K_i, C_j).$$

The overall PPV between two partitionings is obtained by

$$\mathrm{PPV}(K, C) = \frac{\sum_{j=1}^{n} \mathrm{PPV}(C_j) \cdot |T_{\bullet j}|}{|T|}.$$

Definition 2.3.9. Sensitivity The *sensitivity* (Sn) between a clustering C and a reference cluster K_i is defined as the ratio between the correct assignments (TP) and the number of objects of the reference cluster,

$$\mathrm{Sn}(K_i) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \max_j \frac{t_{ij}}{|K_i|}$$

Analogous to the overall PPV, this reference-cluster-wise sensitivity can be combined to an overall sensitivity between two clusterings as

$$\mathrm{Sn}(K, C) = \frac{\sum_{i=1}^{m} |K_i| \cdot \mathrm{Sn}(K_i)}{|K|}$$

Definition 2.3.10. Accuracy The *accuracy* (Acc) is defined as the geometric mean of PPV and sensitivity:

$$\mathrm{Acc}(K, C) = \sqrt{\mathrm{Sn}(K, C) \cdot \mathrm{PPV}(K, C)}$$

One of the most commonly used external quality measures is the so-called F-measure. The original F-measure is defined as the harmonic mean between the general definitions of precision $\left(\frac{\mathrm{TP}}{\mathrm{TP+FP}}\right)$ and recall $\left(\frac{\mathrm{TP}}{\mathrm{TP+FN}}\right)$. This definition must be generalized for usage with clusterings.

Definition 2.3.11. F-measure The F-measure between a clustering C and a reference cluster K_i is defined as

$$\mathrm{F}(K_i, C) = \max_{C_j \in C} \frac{2 \cdot t_{ij}}{|C_j| + |K_i|}.$$

The overall F-measure for the entire clustering is calculated as

$$\mathrm{F}(K, C) = \frac{1}{\sum_{K_i \in K} |K_i|} \sum_{K_i \in K} (|K_i| \cdot \mathrm{F}(K_i, C)).$$

There are also variants of the F-measure where there is a weighting of precision and recall to favor one over the other. The F-measure presented here is also sometimes referred to as F_1-score, meaning neither precision nor recall is favored.

2.3.3. Summary

All measures are summarized in Table 2.1. It is important to acknowledge that there is no superior quality measure. Each individual quality measure emphasis certain properties of the clustering. The internal measure can basically assess two important properties of a clustering [73]:

Compactness "It measures how closely related the objects in a cluster are" [73].

Separation "It measures how distinct or well-separated a cluster is from other clusters." [73].

All internal quality measures presented here, attempt to balance between compactness and separation. As already mentioned, the Dunn index is very susceptible to noise, whereas the silhouette value and the Davies-Bouldin index are more consistent in presence of noise. Furthermore, the silhouette value has the advantage that "the entire clustering is displayed by combining the silhouettes into a single plot, allowing an appreciation of the relative quality of the clusters and an overview of the data configuration" [103] which may help researchers to gain further insights in the reported result.

Generally, whenever a gold standard is available, external measures should be favored over internal measures because only they allow an "entirely objective evaluation and comparison of clustering algorithms on benchmark data, for which the class labels are known to correspond to true cluster structure" [59]. As described in the previous subsection, the external quality measures usually measure combinations of TP, FP, TN, and FN. Depending on the actual clustering task, the external measure emphasizing the most important requirements to the clustering should be chosen. For example, if it is crucial for the cluster quality to minimize the number of false positives, e.g., the Rand index should be preferred over the Jaccard index (the Jaccard index does not account for false positives at all).

Probably the most commonly used measure is the F-measure, as it provides a compromise of precision and recall, both of which are fundamental properties of a clustering. Throughout the thesis, mostly the F-measure is used in order to compare a clustering against a gold standard. On the other hand, it must be emphasized that there is no "best" measure; the choice of measure always depends on the context of the cluster study. It is even harder to pick a good internal measure as they naturally lack a basis to compare with. The researcher necessarily already needs a hint as

to how an optimal clustering should look in order to choose the internal measure emphasizing that property best.

2.4. Clustering Algorithms

Clustering has been a long standing problem in computer science for the unsupervised analysis of datasets. As a result, many different approaches have been developed and published. As different as these approaches are, all of them share common traits. All clustering methods require at least one density parameter which more or less directly influences the average size of the clusters. This necessity is apparent as the algorithm has no means to "know" in advance if the user is looking for a fine or a coarse clustering of the dataset. For example, identifying protein families (finer clustering) or protein super-families (coarser clustering) are both legitimate goals for the same dataset but require different parameters. Furthermore, all clustering algorithms necessarily require a proximity measure for all objects in question.

As already mentioned in the introduction, there is no generally accepted definition of a cluster. Each algorithm is based on the authors' idea of an ideal clustering, often formulated as an objective function which is then maximized (or minimized) by the algorithm. (Estivill-Castro argues in [46] that this is in fact the reason why so many clustering algorithms are developed.) For example, K-means seeks to minimize the average distance of the objects to their cluster centers (minimizing the within-cluster (WC) squared-error criterion, refer to Subsection 2.4.6 on page 46 for more details on K-means), producing mostly spherical clusters [89]. This reason renders it impossible to give general advice as to what clustering tool performs best in terms of cluster quality. For example, if a dataset contains concave and elongate clusters, K-means is a poor choice as the structure of the dataset conflicts with the definition of a cluster on which K-means is based. This section gives a non-exhaustive overview of the most-commonly used clustering algorithms in bioinformatics. We briefly discuss the core principle of each algorithm and the required parameters. The algorithms are ordered alphabetically and neither reflect the importance nor the quality of the algorithm. In the summary (Subsection 2.4.11 on page 53) we give an overview which desirable features are fulfilled by the different clustering tools.

2.4.1. Affinity Propagation (AP)

Affinity propagation [49] performs the clustering by identifying *exemplars* (sometimes also called *prototypes*) among the available data points and reports their neighborhoods as clusters. In contrast to many other exemplar finding clustering methods (sometimes also called prototype methods), affinity propagation considers all data points as possible exemplars. The final set of exemplars is determined by sending messages between the data points voting for the best set of exemplars for the given similarity function in combination with the preference.

Affinity propagation requires as input a similarity function $s(u, v)$ with $u, v \in V$ and $u \neq v$. As a parameter, AP requires the "self-similarity" $s(u, u)$ for each data

Internal Measure	**Formula**
Dunn index	$D(C) = \min\limits_{C_i \in C} \left\{ \min\limits_{\substack{C_j \in C \\ C_i \neq C_j}} \left\{ \frac{\min_{u \in C_i; v \in C_j} d(u,v)}{\max_{C_p \in C} \{\max_{u,v \in C_p} d(u,v)\}} \right\} \right\}$
Davies-Bouldin index	$\mathrm{DB}(C) = \frac{1}{\lvert C \rvert} \sum\limits_{C_i \in C} \left(\max\limits_{\substack{C_j \in C \\ C_i \neq C_j}} \frac{\sigma_i + \sigma_j}{m(\overline{C}_i, \overline{C}_j)} \right)$ with $\sigma_i = \sqrt[p]{\frac{1}{\lvert C_i \rvert} \sum_{c \in C_i} \lvert c - \overline{C}_i \rvert^p}$
Silhouette value	$s(C) = \frac{1}{N} \sum\limits_{u \in V} \frac{b(u) - a(u)}{\max\{a(u), b(u)\}}$
External Measure	
Rand index	$R(K,C) = \frac{TP+TN}{\binom{N}{2}}$
Jaccard index	$J(K,C) = \frac{TP}{TP+FP+FN}$
Overlap score	$\omega(K_i, C_j) = \frac{\lvert K_i \cap C_j \rvert^2}{\lvert K_i \rvert \lvert C_j \rvert}$
Positive predictive value	$\mathrm{PPV}(K,C) = \frac{\sum_{j=1}^{n} \left(\max_i \left(\frac{t_{ij}}{\lvert T_{\bullet j} \rvert} \right) \right) \cdot \lvert T_{\bullet j} \rvert}{\lvert T \rvert}$
Sensitivity	$\mathrm{Sn}(K,C) = \frac{\sum_{i=1}^{m} \lvert K_i \rvert \cdot \left(\max_j \frac{t_{ij}}{\lvert K_i \rvert} \right)}{\lvert K \rvert}$
Accuracy	$\mathrm{Acc}(K,C) = \sqrt{\mathrm{Sn}(K,C) \cdot \mathrm{PPV}(K,C)}$
F-measure	$\mathrm{F}(K,C) = \frac{1}{\sum_{K_i \in K} \lvert K_i \rvert} \sum\limits_{K_i \in K} \left(\lvert K_i \rvert \cdot \left(\max_{C_j \in C} \frac{2 \cdot t_{ij}}{\lvert C_j \rvert + \lvert K_j \rvert} \right) \right)$

Table 2.1.: Overview of the most commonly used measures. $C = \{C_1, \ldots, C_N\}$ denotes the clusters reported by the algorithm, $K = \{K_1, \ldots, K_2\}$ the gold standard, $\overline{C}_i$ the centroid of the ith cluster. $d(u,v)$ represents the distance measure. The p in the Davis-Bouldin index refers to the same p as in the used Minkowski distance. $T = (t_{ij})$ is the confusion matrix with $|T|$ being the sum of all entries, $|T_{\bullet j}|$ as the sum of all entries in jth column, and $|T_{i\bullet}|$ as the sum of all entries in the ith row. In the silhouette value, $a(u)$ is the average distance of x to all objects in its own cluster, $b(u)$ the average distance of u to the closet other cluster. For the Jaccard and Rand index, TP, TN, FP and FN are defined following the pairwise approach.

point u, which determines the probability of u to be selected as exemplar. Usually, all data points get the same value assigned; thus, AP requires only one parameter to be set. During a run of AP, two different types of messages are exchanged, namely:

Responsibility The responsibility $r(u, v)$ is a number that is sent from point u to candidate exemplar v, reflecting how well suited v is as an exemplar for u.

Availability The availability $a(u, v)$ is a number that is sent in the opposite direction, from candidate exemplar v to point u. The availability corresponds to how feasible it is for point u to choose v as exemplar.

The authors state that both messages can be seen as the log-probability ratios for point v being an exemplar and u choosing v as its exemplar [49]. The algorithm starts by setting the availability to zero,

$$a(u, v) = 0 \quad \forall u, v \in V.$$

Afterwards, the responsibilities are updated as follows:

$$r(u, v) = s(u, v) - \max_{u \neq w} \{a(u, w) + s(u, w)\}.$$

The responsibility of v for the point u is highest when v only weakly prefers different points as its exemplar. A negative self-responsibility means that the point rather belongs to a different exemplar than being an exemplar itself. In the next step, the availability of u, v with $u \neq v$, is updated to

$$a(u, v) = \min \left\{ 0, r(v, v) + \sum_{w \neq u, v} \max\{0, r(w, v)\} \right\}$$

whereas the self-availability is set to

$$a(v, v) = \sum_{u \neq v} \max\{0, r(u, v)\}.$$

That means, the more responsible v is for other points w, the higher is the availability of point u to v. "To limit the influence of strong incoming positive responsibilities, the total sum is thresholded so that it cannot go above zero" [49]. After each iteration, the actual cluster structure is determined by finding the exemplar for every data point u. The exemplar for u is that data point v maximizing the function

$$a(u, v) + r(u, v).$$

If the function is maximized for $i = j$, this means the point x_i is its own exemplar. In order to prevent numerical oscillations, a user-defined dumping factor $\lambda \in [0, 1]$ is introduced. The kth iteration of $r^{(k)}(u, v)$ and $a^{(k)}(u, v)$ is then calculated as

$$\begin{aligned} r^{(k)}(u,v) &= \lambda \cdot r^{(k-1)}(u,v) + (1-\lambda) \cdot r(u,v) \\ a^{(k)}(u,v) &= \lambda \cdot a^{(k-1)}(u,v) + (1-\lambda) \cdot a(u,v). \end{aligned}$$

The algorithm terminates whenever the exemplar decision does not change for a certain number of iterations, the availability/responsibility-changes fall below a certain threshold, or a user-defined number of iterations is reached. Convergence of the algorithm cannot be guaranteed; thus, the termination criteria can also be regarded as additional parameters as the clustering result can change depending on the number of iterations performed.

2.4.2. CFinder

CFinder [3, 92] is an overlapping clustering approach based on discovering k-clique communities. The input is considered a graph of which all maximal cliques (fully connected subgraphs which are not part of a larger fully connected subgraph) of size k or larger are extracted. The authors implemented the clique finding strategy as follows:

1. The maximal possible clique size γ is determined by the degree sequence of the nodes. (A node with degree d cannot be part of a clique $\mathcal{C}_i$ of size $|\mathcal{C}_i| > d$ as each member of a clique requires an edge connecting it to all other members of the clique. The algorithm can only find a clique of size l if there are at least l nodes having a degree of at least $l - 1$.)

2. Next, nodes are repeatably chosen and all cliques of size γ containing the chosen node are extracted. Then, the node and all its adjacent edges are removed from the graph until all cliques of size γ are discovered. The process is now repeated with the new clique size $\gamma_{\text{new}} = \gamma - 1$ on the original graph until $\gamma = k$.

The discovery of cliques of size γ containing the node v is achieved by maintaining two sets of nodes A and B. A is initialized by containing only v, B with all neighbors v. Then the algorithm recursively transfers a node $u \in B$ to A. All nodes of B which are not neighbors of the newly transferred node u are removed; thus, B contains only those nodes which are directly connected to all nodes in A. Whenever A reaches the size of γ, the clique is reported. The recursion ensures that all combinations of node-transfers from B to A are tested and thus, all cliques of size γ containing v are found.

After all L cliques $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_L\}$ of size at least k are found, each clique $\mathcal{C}_i$ is now considered a node in an adjacency graph $G = (\mathcal{C}, E)$. The edges E of the graph G fulfill the following:

$$\mathcal{C}_i\mathcal{C}_j \in E \quad \Longleftrightarrow \quad |\mathcal{C}_i \cap \mathcal{C}_j| \geq k - 1 \qquad \forall \mathcal{C}_i, \mathcal{C}_j \in \mathcal{C}.$$

The connected components of this adjacency graph now form the k-clique communities. All nodes in one of these connected components form a cluster in the final result. The user can specify k which is suggested by the authors to be between four and six [3].

2.4.3. Clustering with Overlapping Neighborhood Expansion (ClusterONE)

Clustering with Overlapping Neighborhood Expansion (ClusterONE) [87] is a recently developed clustering method for detecting overlapping protein complexes. The primary concept behind the algorithm is the so-called cohesiveness measure, which is a combination of the number of reliable interactions within a cluster and the separation of the cluster from the rest of the network. Letting U be a group of nodes, the cohesiveness $f(U)$ is given by

$$f(U) = \frac{w^{\text{in}}(U)}{w^{\text{in}}(U) + w^{\text{bound}}(U) + p|U|}.$$

Here, $w^{\text{in}}(U)$ denotes the sum of all edge weights within U; $w^{\text{bound}}(U)$ represents the sum of all edge weights connecting this group with the rest of the network. p is used to express the uncertainty within the data, i.e., the probability that there are yet undiscovered connections from the group to the network.

The algorithm starts with growing cohesive groups from a seed node. The node u with the highest degree is selected as first seed node. Let $U^{(0)} = u$ be the initial set and let $U^{(t)}$ be the cohesive group in iteration t. The set $U^{(t+1)}$ is determined by either adding the incident external vertex $v \notin U^{(t)}$ yielding the greatest gain of $f(U^{(t)} \cup \{v\})$, or by removing a previously added (and now suboptimal) internal node $w \in U^{(t)}$ yielding the greatest gain of $f(U^{(t)} \backslash \{w\})$. The growth-process terminates whenever no vertex for neither adding nor removing can be discovered to increase $f(U^{(t)})$. The next cohesive group is seeded by the yet unassigned node with the highest degree until no unassigned nodes are left. A node can only be a seed node once (i.e., when the seed node gets removed from the group during the growth process).

After this growth process, cohesive groups with an overlap score $\omega > 0.8$ are merged into one group (for the overlap score, refer to definition 2.3.6 on page 36).

In a last step, all clusters with less than three objects and loosely connected clusters whose density

$$\frac{w^{\text{in}}(U)}{\binom{|U|}{2}}$$

is below a given threshold δ, get removed from the reported clustering result.

2.4.4. Clustering Based on Maximal Cliques (CMC)

The Clustering based on Maximal Cliques (CMC) [72] also operates on a graph representation of the input data. CMC was especially designed to work on unweighted,

i.e., binary interaction networks of proteins. The main idea is to iteratively reweigh the edges of the binary network before maximal cliques are extracted.

Let $w^{(t)}(u,v)$ denote the weight of the edge uv in the tth iteration. The weights of $w^{(0)}(u,v)$ are set to either 1 or 0, depending upon whether the edge does or does not exist. Furthermore, let N_u denote the neighborhood of node u. The iterative reweighing is performed by

$$w^{(t)}(u,v) = \frac{\sum_{w \in N_u \cap N_v} \left(w^{(t-1)}(w,u) + w^{(t-1)}(w,v) \right)}{\sum_{w \in N_u} w^{(t-1)}(w,u) + \lambda^{(t)}(u) + \sum_{w \in N_v} w^{(t-1)}(w,v) + \lambda^{(t)}(v)} \quad \text{with}$$

$$\lambda^{(t)}(u) = \max \left\{ 0, \frac{\sum_{v \in V} \sum_{w \in N_v} w^{(t-1)}(v,w)}{|V|} - \sum_{w \in N_u} w^{(t-1)}(w,u) \right\}.$$

Basically, $w^{(t)}(u,v)$ is the weight of the connections of the common neighborhood of the points u and v divided by the weight of the entire neighborhood of u and v. The correction factor $\lambda^{(t)}(u)$ lifts weakly connected nodes of the network (when the sum of the weights of the adjacent edges of u is below the network average) by lifting them to at least the average neighborhood weight. The authors state that after two iterations the clustering result does not improve anymore.

In the next step, all maximal cliques are extracted. Even though extracting maximal cliques of graphs is NP-hard, it does not pose a problem as PPI networks are sparse. However, this can lead to problems, when the algorithm is applied to different clustering problems. The extracted cliques $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_L)$, which can have an large overlap, are post-processed in order to merge them into highly connected clusters. In order to do so, each extracted clique $\mathcal{C}_k$ is scored by

$$\text{score}(\mathcal{C}_k) = \frac{\sum_{u,v \in \mathcal{C}_k} w(u,v)}{|\mathcal{C}_k| \cdot (|\mathcal{C}_k| - 1)}.$$

Here, $w(u,v)$ is normally the value of the last iteration of the reweighing. Furthermore, the inter-clique score between two cliques $\mathcal{C}_k$ and $\mathcal{C}_l$ is defined as

$$\text{inter-score}(\mathcal{C}_k, \mathcal{C}_l) = \sqrt{\frac{\sum_{u \in \mathcal{C}_k \setminus \mathcal{C}_l} \sum_{v \in \mathcal{C}_l} w(u,v)}{|\mathcal{C}_k - \mathcal{C}_l| \cdot |\mathcal{C}_l|} \cdot \frac{\sum_{u \in \mathcal{C}_l \setminus \mathcal{C}_k} \sum_{v \in \mathcal{C}_k} w(u,v)}{|\mathcal{C}_l - \mathcal{C}_k| \cdot |\mathcal{C}_k|}}.$$

Let $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_L)$ be the extracted cliques sorted by their score in descending order. For every clique $\mathcal{C}_k$, a clique C_l with a lower score is searched such that $\frac{|\mathcal{C}_k \cap \mathcal{C}_l|}{|\mathcal{C}_l|} \geq t_o$, with t_o being the user-defined "overlap threshold". If such a $\mathcal{C}_l$ is found and inter-score$(\mathcal{C}_k, \mathcal{C}_l) \geq t_m$, with t_m being the user-defined "merge threshold", both cliques are merged. If the inter-score is below t_m, $\mathcal{C}_l$ is removed from the list of cliques. After the merging and removing process, the remaining cliques are returned as the final clustering result.

2.4.5. Hierarchical Clustering

Hierarchical clustering approaches are different from the other clustering methods presented in this section. In contrast to the other methods, hierarchical clustering does not return a partitioning of the input object set into a set of clusters. Rather they return a sequence of consecutive partitionings represented as a tree, ranging from one big cluster containing all objects, to the point where every object is a singleton. The different approaches can be divided into two main branches [47]:

- *Divisive* or *top-down*: The clustering starts with one big clustering containing all objects and subsequently divides the cluster into finer parts.
- *Agglomerative* or *bottom-up*: This is the opposite approach starting with all objects being singletons. These singletons are merged into coarser clusterings with each step of the process.

There exist an enormous number of approaches to calculate the division- or agglomeration-points. A good overview can be found in [55]. Here, we will focus on the most commonly used agglomerative methods. The clustering process starts with a partitioning of the dataset $V = \{v_1, \ldots, v_N\}$ in only singletons. In subsequent steps, the two most similar clusters are merged into one cluster. Therefore, a similarity $s(C_i, C_j)$ between two clusters C_i, C_j must be defined. Furthermore, let $\mathrm{sim}(u, v)$ be the similarity between two objects $u, v \in V$ with $u \neq v$. The most commonly used approaches are:

Single linkage The single linkage method is also called *nearest neighbor*. This method does not take the cluster structure into account and tends to produce elongated clusters [47].

$$s(C_i, C_j) = \max \{\mathrm{sim}(u, v); \quad u \in C_i, v \in C_j\}$$

Complete linkage This cluster similarity measure is also called *furthest neighbor*. In contrast to the "nearest neighbor" this methods prefers compact clusters with equal diameter [47].

$$s(C_i, C_j) = \min \{\mathrm{sim}(u, v); \quad u \in C_i, v \in C_j\}$$

Average linkage This method tends to join clusters with small variance and can be seen as a compromise between nearest and furthest neighbor [47].

$$s(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{u \in C_i} \sum_{v \in C_j} \mathrm{sim}(u, v)$$

As hierarchical clustering produces a series of clusterings, in the agglomerative case ranging from only singletons to one cluster containing all objects, a certain cut-off must be defined for receiving a partial clustering. As examples, that cut-off can be a

certain hierarchy level, or more complex version using nodes of the tree of different levels as clusters, e.g., the dynamic tree cutting [70]. The definition of the cut can be seen as the density parameter for hierarchical clusterings.

All hierarchical clusterings suffer from a common problem: The division of agglomeration performed at a certain step cannot be undone in a later step, even though that would yield to a better overall clustering. In other words, the decision is normally made locally, meaning that it looks greedily for the best possible split/agglomeration of the current clustering and disregards the consequences for the subsequent steps.

2.4.6. K-means

K-means [75] clustering is probably the most well-known clustering method and has already been around for several decades. K-means' general idea is the separation of the dataset V into k disjoint classes $C = \{C_1, \ldots, C_k\}$, such that the distance of all points to the center of their class gets minimized. Let μ_i denote the center of cluster C_i. Thus the optimal partitioning $C' = \{C'_1, \ldots C'_k\}$ is that partitioning which minimizes the object function

$$C = \underset{C'}{\operatorname{argmax}} \left(\sum_{C'_i \in C'} \sum_{v \in C'} d(v, \mu_i)^2 \right)$$

with $d(v, \mu_i)$ being a distance function. As the explicit enumeration of all possible separations of V into k groups is infeasible, heuristics are applied.

The most common approach is the so-called "Least Squares Quantization" of S. Lloyd, developed in 1957, published in 1982 [74]. The algorithms starts with randomly chosen initial cluster centers. In the second step, all points are assigned to that cluster whose mean is closest to the particular point. Based on the current assignments of the points to their clusters, new cluster centers are calculated. This process of assignment and refinement is repeated until the assignment of points to clusters does not change anymore. Naturally, such a heuristic can be stuck in a local minimum; thus, the initial selection of the clustering centers is of great importance.

Most currently conducted research is dedicated to specify the number of clusters and to optimize the initial choice of clustering centers. Note, that when using a distance function like the Euclidean distance, the cluster centers can easily be calculated as that point minimizing the distance to all points of the cluster. That center is not required to be an element of the dataset V. An adaption of K-means, K-medoids, is applied to instances where such a center cannot be calculated. In these cases, μ_i can for example be chosen as $\mu_i = \underset{v \in C_i}{\operatorname{argmin}} \sum_{u \in C_i \setminus \{v\}} d(u, v)$.

2.4.7. Markov Clustering (MCL)

In Markov Clustering (MCL) [42, 112], the problem instance is again interpreted as a graph. The graph is represented as a column stochastic matrix

$$M = (m_{ij}) \in [0,1]^N; \quad \sum_{j=1}^{N} m_{ij} = 1 \quad \forall 1 \leq i \leq N.$$

Random walks on the input graph are now simulated by the repeated alternating application of the following two operations:

Expansion The expansion step is obtained by squaring the matrix

$$M = M^2.$$

The authors state, this step simulates long walks as "it associates new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination" [42]. After that step, the matrix M is no longer column stochastic. This is repaired in the next step.

Inflation The second step simulates random walks within each cluster, thus enforcing the gradual separation of the clusters. This is done by taking each entry of the matrix to the power of r (the user-defined inflation parameter) and scaling the matrix back to a column stochastic matrix:

$$m_{ij} = \frac{m_{ij}^r}{\sum_{i=1}^{N} m_{ij}^r}.$$

The parameter r hugely influences the final clustering, as higher values for r result in more densely connected smaller clusters. The final clustering is obtained by returning the connected components represented by the matrix M. In practice, the algorithm converges quite quickly after ten iterations to a stable state.

2.4.8. Restricted Neighborhood Search Clustering (RNSC)

The Restricted Neighborhood Search Clustering (RNSC)[66] is a randomized graph-based clustering approach. It performs a cost-based local search starting with a random assignment of nodes to clusters. In each iteration, nodes are randomly shuffled between clusters in order to improve the overall cost function based on the number of intra-cluster and inter-cluster edges. Given a clustering $C = (C_1, \ldots, C_K)$ of the object set V, the overall goal is to minimize the following "scaled cost function":

$$\text{cost}(G, C) = \frac{|V| - 1}{3} \sum_{v \in V} \frac{\alpha_v}{\beta_v}.$$

Here, α_v is the number of "bad connections" of node v, i.e., the number of connections of v to nodes not elements of the same cluster of v. Letting $v \in C_i$ and N_v the

neighborhood of v, then $\alpha_v = |N_v \backslash C_i \cup C_i \backslash N_v|$. β_v is a measure for the influence of the node v, how many nodes are influenced by v. Again, let $v \in C_i$, then $\beta_v = |N_v \cup C_i|$. In order to increase the computational speed, RNSC also uses the integer based "naive cost function"

$$\text{cost}_{\text{naive}}(G, C) = \frac{1}{2} \sum_{v \in V} \alpha_v.$$

Both cost-functions reach their lowest values with clusterings with a high degree of connectivity within a cluster and a low connectivity between two clusterings.

Generally, the algorithm starts off with a random clustering C_r and tries to improve the naive costs of that clustering by moving nodes randomly from one cluster into another. If the costs of the clustering are not improved over the last $\mathcal{T}_n$ steps (the user-defined parameter "naive stopping tolerance"), this phase ends.

The best clustering found is now optimized in order to improve the scaled cost function using $\mathcal{L}_E$ steps (the user-defined parameter "scaled experiment length"). Additionally to the random moves, during the optimization of the scaled cost function, every $\mathcal{F}_\mathcal{D}$-steps (the user-defined parameter "diversification period") $\mathcal{L}_D$ (the user-defined parameter "diversification length") diversifications steps are performed. In this step, a randomly selected cluster is destroyed and all elements are distributed to different clusters. In order to improve the quality of the clustering results, RNSC maintains a tabu-list of length $\mathcal{L}_T$ (user-defined parameter "tabu length") which stores nodes which are not allowed to be moved anymore since they did not improve the result. The parameter $\mathcal{T}_T$ "tabu tolerance" allows nodes to appear several times in the tabu list and they are classified as unmovable when they are $\mathcal{T}_T$-times in the list.

The best clustering achieved after both phases is returned as the result of that "experiment". As RNSC is a randomized algorithm, $\mathcal{N}_E$ (user-defined parameter "number of experiments") such runs are performed until the final result is reported to the user. Furthermore, the parameter $\mathcal{N}_C$, "maximal number of clusters" limits the number of allowed clusters to $\mathcal{N}_C$ which is comparable to a density parameter.

2.4.9. Repeated Random Walks (RRW)

The Repeated random walks (RRW)[76] algorithm detects clusters by simulating random walks on the given input graph with restarts. The graph is represented as a row normalized matrix

$$M = (m_{ij}) \in [0, 1]^N; \quad \sum_{i=1}^{N} m_{ij} = 1 \quad \forall 1 \leq j \leq N.$$

The algorithm starts with calculating random walks for every node $v_i \in V$. The result is a stationary vector $s(v_i) = \{\sigma_1, \ldots, \sigma_N\}$ which indicates the proximity of

node v_i to node v_j as σ_j. The starting point for the iterative process is $s^{(0)}(v_i)$ with all entries set to zero except σ_i set to 1. The kth step is calculated with

$$s^{(k)}(v_i) = \alpha s^{(0)}(v_i) + (1-\alpha) M^T s^{(k-1)}(v_i).$$

This process stops whenever $s^{(k)}(v_i)$ converges. The parameter α is the user-defined restart probability. With a larger α, the random walk is concentrated to smaller walks resulting in smaller clusters. The vectors are the basis for calculating random walks starting from a cluster instead of a single node. The random walk for an entire cluster C_k is simulated by the linear combination of the stationary vectors $s(v_i)$ of all nodes of the cluster:

$$s(C_k) = \frac{1}{|C_k|} \sum_{v \in C_k} s(v).$$

In order to build the clusters, the algorithm starts with each node as a cluster and repeatedly expands that cluster. Letting $\mathcal{C}_i^{(0)} = \{v_i\}$ be the cluster to expand, and $sc^{(k)}$ be the proximity of the point to the cluster added in the kth step ($sc^{(0)} = 0$).

$$\mathcal{C}_i^{(k)} = \mathcal{C}_i^{(k-1)} \cup \operatorname{argmax}\left\{ s(\mathcal{C}_i^{(k-1)}) \right\} \quad \Longleftrightarrow \quad \max\left\{ s(\mathcal{C}_i^{(k-1)}) \right\} \geq \lambda \cdot sc^{(k-1)},$$

with $\lambda \in [0,1]$ being the "early cutoff", the function $\operatorname{argmax}\left\{ s(\mathcal{C}_i^{(k-1)}) \right\}$ returns that point with the highest proximity to the cluster $\mathcal{C}_i^{(k-1)}$. The process stops whenever either no point exceeds the early cutoff λ, or the cluster whenever the cluster reaches the user-defined maximal cluster-size of K. All clusters found are now sorted by their statistical significance which is calculated by

$$\text{p-value}(\mathcal{C}_i) = 1 - \overline{\mathcal{C}_i} \cdot \sqrt{|\mathcal{C}_i|}.$$

Here, $\overline{\mathcal{C}_i}$ denotes the average proximity of all points of the cluster $\mathcal{C}_i$ to each other. In a final step, for all overlapping clusters $\mathcal{C}_i, \mathcal{C}_j$ with an overlap score $\omega(\mathcal{C}_i, \mathcal{C}_j) \geq \gamma$, with γ being the user defined overlap threshold, the cluster with the smaller significance is removed (Note that the authors use a slightly different definition of the overlap score: $\omega(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i \cap \mathcal{C}_j|}{\min\{|\mathcal{C}_i|, |\mathcal{C}_j|\}}$). All remaining clusters are reported as the final clustering result.

2.4.10. Transitivity Clustering (TransClust)

Transitivity Clustering (TransClust) [122, 123] is based on the *weighted transitive graph projection problem* (WTGPP), also called the cluster editing problem. The input is interpreted as a graph. For a user-given threshold t, we can define a threshold graph $G_t = (V, E_t)$ with

$$E_t = \{uv : s(u,v) > t\}.$$

In other words, the graph contains only those edges whose similarity value exceeds the user-specified threshold.

The resulting graph usually decomposes into several connected components, which could be already used as a clustering, i.e., every connected component corresponds to a cluster. Nevertheless, this method potentially matches very dissimilar elements into one cluster whenever there is a path between those elements. In order to account for that fact, we transfer the graph into a transitive graph $G' = (V, E')$ by adding or removing edges. For each editing step, we assign corresponding edit costs of $\text{cost}(uv) = |s(u,v) - t|$ for adding or removing the edge uv. The total editing costs for transferring the graph G_t into the transitive graph G' can be calculated by

$$\text{cost}(G_t \to G') = \underbrace{\sum_{uv \in E_t \setminus E'} |s(u,v) - t|}_{\text{deletion costs}} + \underbrace{\sum_{uv \in E' \setminus E_t} |s(u,v) - t|}_{\text{insertion costs}}.$$

The solution of the WTGPP is given by the graph with the least edit costs. One observation is that each connected component of G_t can be treated as a own problem instance. In terms of total costs it is never beneficial to join two connected components into one cluster, as all edges between these two connected components are below the threshold. Thus, the algorithm only needs to transform each connected component into a transitive graph. Even though that significantly reduces the problem size in most problem instances, it has already been shown that the WTGPP is NP-hard [120] and even APX-hard [120, 28]. Thus, the problem can only be solved exactly for small instances; for larger instances, heuristics are applied.

Greedy heuristic The greedy algorithm CAST [98, 17] starts by removing "the most promising" edges. Afterwards, the the transitive solution is generated by reporting the transitive closure of the remaining connected components as result of the clustering. The main challenge is to identify those edges whose removal yield the greatest gains in the overall costs. Here, the algorithm concentrates on the conflicting triples $uvw \in [V]^3$ ($[V]^3$ denotes the set of all three-element subsets of V). Without loss of generality, let the entire graph $G = (V, E)$ be a connected component and the similarity function s is chosen in such a way that two objects u and v are similar iff $s(u,v) \geq 0$, i.e., a threshold of $t = 0$ is selected. Furthermore, let $\mathfrak{C}(G)$ the set of all conflicting triplets of G. The deviation $D(G)$ of the entire graph G from transitivity is defined as

$$D(G,s) = \sum_{uvw \in \mathfrak{C}(G)} \min\{|s(u,v)|, |s(u,w)|, |s(v,w)|\}.$$

In order to decide, which edge to remove, the transitivity improvement $\text{imp}(uv)$ of the removal of uv is defined as

$$\text{imp}(uv) = D(G) - D(G', s') - s(u,v)$$

with $G' = (V, E \setminus \{uv\})$ and s' as the identical similarity function as s except that $s'(uv) = -\infty$. The algorithm proceeds with removing the highest scoring edges until the graph G decomposes into two connected components G_1 and

G_2. All removed edges which do not reconnect G_1 and G_2 are added back to the edge set of the corresponding subgraph. This procedure is repeated with G_1 and G_2 until all connected components are either transitive or no edge-removal is beneficial. The transitive closure of the connected components are returned as the final clustering result.

Nature-inspired The FORCE heuristic is a nature-inspired graph layout algorithm simulating repulsion and attraction forces of the nodes, similar to the layout algorithm of Fruchterman and Reingold [121]. The process is divided into three steps, the layout and the partitioning phase followed by a post-processing phase. The algorithm starts by arranging the nodes $u \in V$ on a usually two-dimensional plane (higher dimensions are also supported) in a circle with radius ρ. Let $\text{pos}^{(r)}(u) \in \mathbb{R}^2$ denote the current position of u in the pane in iteration r. For each node u, the repulsion or attraction forces $f_{u \leftarrow v}^{(r)}$ inflicted by node v can be calculated by

$$f_{u \leftarrow v}^{(r)} = \begin{cases} \frac{\text{cost}(uv) \cdot f_{\text{att}} \cdot \log(d^{(r-1)}(u,v)+1)}{|V|} & \text{if } s(u,v) \geq t \text{ (attraction)} \\ \frac{\text{cost}(uv) \cdot f_{\text{rep}}}{|V| \cdot \log(d^{(r-1)}(u,v)+1)} & \text{if } s(u,v) < t \text{ (repulsion)} \end{cases}$$

with $d^{(r-1)}(u,v)$ being the Euclidean distance of $\text{pos}^{(r-1)}(u)$ and $\text{pos}^{(r-1)}(v)$ in the plane, and f_{att} and f_{rep} as two parameters defining the influence of the repulsion or attraction forces. The accumulated displacement vector $\Delta_u^{(r)}$ for each point u in the rth iteration is calculated as

$$\Delta_u^{(r)} = \sum_{v \in X \setminus \{u\}} f_{u \leftarrow v} \cdot \frac{\text{pos}^{(r-1)}(u) - \text{pos}^{(r-1)}(v)}{d^{(r-1)}(u,v)}$$

In order to retain a cool-down effect, the maximal displacement in iteration r is limited by $M(r)$. Thus, the new position of u is calculated as

$$\text{pos}^{(r)}(u) = \text{pos}^{(r-1)}(u) + \frac{\Delta_u^{(r)}}{||\Delta_u^{(r)}||} \cdot \min\{||\Delta_u^{(r)}||, M(r)\}.$$

In this algorithm, all parameters are optimized by utilizing the parameter set resulting in the least edit costs [121].

In the next phase, the points are assigned to clusters by a geometric single-linkage clustering with a maximal node distance δ. The algorithm begins with an arbitrary node u to build the cluster $\mathcal{C}_i^{(0)}$ by assigning u to that cluster. In the next steps, $C_i^{(r-1)}$ gets expanded with

$$\mathcal{C}_i^{(r)} = \mathcal{C}^{(r-1)} \cup \{v; \exists w \in \mathcal{C}_i^{(r-1)} : d(v,w) < \delta\}.$$

Again, δ can be optimized by choosing the δ resulting in the least edit costs.

Let $C = \{C_1, \ldots, C_L\}$ be the clustering obtained by the partitioning phase. The algorithm performs two different post-processing optimizations on that clustering C:

1. Merge Clusters: The algorithm merges a pair of clusters C_i, C_j and keeps the clustering $C' = \{C_1, \ldots, C_i \cup C_j, \ldots, C_L\}$ whenever $\text{cost}(C') < \text{cost}(C)$. This process is repeated with C' as a new reference clustering C until no pair of clusters can be found which further reduces the costs.
2. Move nodes: Now, let C be the already optimized clustering of step 1. In this step, a node $u \in C_k$ is picked and assigned to a different cluster C_l. If the clustering $C' = \{C_1, \ldots, C_k \backslash \{u\}, \ldots, C_l \cup \{u\}, \ldots, C_L\}$ yields to no cost-reduction, another target cluster C_l with $l \neq k$ is selected. If no assignment of u to a different cluster improves the result, the next point is selected. Whenever such an exchange is beneficial with respect to the costs, the process starts over with C' as a new reference clustering C.

After this optimization, the final result is returned.

Exact FPT The first fixed parameter tractable (FPT) algorithm for the WTGPP was introduced by Rahmann *et al.* in 2007 [98]. The parameter k used is the maximal cost of the solution. Again, without loss of generality, let the similarity function s be chosen in such a way that two objects u, v are similar iff $s(u, v) \geq 0$; thus, a threshold of $t = 0$ is selected. All non-transitive connected components are treated by the algorithm as an own problem instance. The first step is to "check for unaffordable edge modifications" in the connected component $G = (V, E)$ in order to reduce the problem size. For each edge $uv \in \binom{V}{2}$, the lower bounds

$$\begin{aligned} \text{icf}(uv) &= \sum_{w \in N_u \cup N_v} \min\{s(w, u), s(w, v)\} \\ \text{icp}(uv) &= \sum_{w \in N_u \Delta N_v} \min\{|s(w, u)|, |s(w, v)|\} \end{aligned}$$

are calculated. Here, $\text{icf}(uv)$ denotes the minimal costs for removing the edge uv (i.e., putting u and v in different clusters) and $\text{icp}(uv)$ the minimal costs for keeping the edge uv (i.e., putting u and v in the same cluster). The set $N_u \cup N_v$ is the common neighborhood of u and v (if u and v should be put into different clusters, every common neighbor needs to be assigned to one of the two clusters) and $N_u \Delta N_v$ is the symmetric set difference (if u and v should be put into the same cluster, each non-common neighbor must be put in either the same cluster as u and v or into a different one). When $\text{icf}(uv) > k$, the edge uv is set to *permanent*, when $\text{icp}(uv) > k$ the edge is set to *forbidden*. If both lower bounds are greater than k, the problem cannot be solved for the given maximal k. When all pairs of nodes uv connected by an edge marked as permanent are merged into a new node w, the costs for all operations for edges

incident to w are calculated as a combination of the costs for the nodes u and v. The parameter k is reduced by the costs used for performing the merging.

The Branching strategy iterates over all conflicting triplets $uvw \in \binom{X}{3}$ with u being the node with a degree of two and distinguishes between three different branches:

- Insert vw and set all edges to *permanent*
- Delete uv, set uw to *permanent*, and set both other edges to *forbidden*
- Delete uw and set only uw to *forbidden*

With every operation, the parameter k is reduced by the according costs. If an operation exceeds k, the branch can be neglected.

TransClust operates using all mentioned algorithms. For every connected component, the upper-cost limit is calculated by the greedy heuristic. If this upper bound is low enough, TransClust solves that instance with the exact FPT. In all other cases, the FORCE heuristic is applied.

2.4.11. Summary

In the previous subsections, we introduced widely-used clustering approaches in bioinformatics. The projects' websites and the according citations are summarized in Table 2.2. As already mentioned, "there is no single approach to clustering that can be regarded as appropriate for most situations" [83], thus we do not attempt to rank the previously introduced clustering tools according to a certain performance measure. In other words, the performance of a clustering tool can only be judged in combination with a certain dataset, thus in a "method × data" relation [47]. ClustEval, which will be introduced later in Chapter 6 on page 107, eases conducting comparative cluster analyses and we will give a performance comparison of various tools on different datasets.

In this subsection we give an overview on what we deem to be generally desirable characteristics of a clustering tool and tabulate the presence of absence of these characteristics for a selection of tools. The following list of the characteristics is based on the work of Andreopoulos *et al.* [7] and Wittkop [120]. This selection reflects our experience with cluster analyses and is not intended to be exhaustive or universally accepted. Depending on the actual clustering task, certain characteristics may gain or lose importance or even non-mentioned features may become crucial. The characteristics we consider important in current-day cluster analyses of biomedical data are:

Scalability: The ability to perform well in terms of runtime and memory consumption in both small and large datasets.

In the framework of this thesis, we require a clustering algorithm to be able to cope with datasets most prevalent in bioinformatics on a standard computer. We consider

	Website	Availability	Citation
Affinity Propagation	http://www.psi.toronto.edu/index.php?q=affinity%20propagation	online, free	[49]
CFinder	http://cfinder.org/	upon request	[3, 92]
ClusterONE	http://www.paccanarolab.org/cluster-one/index.html	online, free	[87]
CMC	http://www.comp.nus.edu.sg/~wongls/projects/complexprediction/CMC-26may09/	online, free	[72]
Hierarchical Clustering	different implementations	libraries, free	-
K-means	different implementations	libraries, free	[74]
Markov Clustering	http://www.micans.org/mcl/	online, free	[42, 112]
RNSC	http://www.cs.utoronto.ca/~juris/data/rnsc/	upon request	[66]
RRW	http://cs.ucsb.edu/~kpm/RRW/	online, free	[76]
TransClust	http://transclust.mpi-inf.mpg.de	online, free	[122, 123]

Table 2.2.: Overview over the existing clustering tools typically utilized in bioinformatics and the project websites. There is no reference implementation for neither K-means or hierarchical clustering. These tools can be obtained as libraries for many different programming languages or they are included in statistical software packages like WEKA [58].

	Affinity Propagation	CFinder	ClusterONE	CMC	Hierarchical Clustering	K-means	Markov Clustering	RNSC	RRW	TransClust
Scalability	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
Robustness	✓	✗	✓	✗	✗	✗	✓	✗	✓	✓
Integration of existing knowledge	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
Arbitrary-shaped clusters	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Minimal user-specified input	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Interpretable results	✗	(✓)	✗	✗	✗	✓	✗	✗	✗	✓
Reproducibility of results	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓
Missing Values	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Active Clustering	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 2.3.: Not all currently used clustering tools in bioinformatics support all features defined in in the text. This table provides an overview of the characteristics of the different tools.

the size of a typical dataset to range from a couple of hundreds of objects (e.g., gene expression datasets) to 100,000 objects (e.g., sequence datasets).

Robustness: "Ability to detect outliers that are distant from the rest of the samples" [7] and the ability of the algorithm to produce consistent results in presence of noise.

As the data we consider in this thesis usually originates from wet-lab experiments and the measurements often contain noise in terms of small errors due to the measurement procedure (technical errors). Thus, a clustering algorithm should produce meaningful results even in the presence of noisy data and produce similar results with slightly different input. Nevertheless, the algorithm should be able to group distant outliers (caused by natural variations in the dataset) into a group of otherwise homogeneous objects [7].

Integration of existing knowledge: The ability of the algorithm to incorporate user-provided *a priori* knowledge of the clustering.

In many cases, the researcher has *a priori* knowledge about the optimal clustering. This may be, for instance, the information that certain objects have to be grouped in one cluster or in two separate clusters. The clustering algorithms should be capable of integrating such information in order to help reducing false positives and false negatives (refer to Subsection 2.3.2 on page 35 for the definitions of false positive and false negative in the context of clustering).

Arbitrary-shaped clusters: "A clustering algorithm should find arbitrary shaped clusters" [7].

Typical cluster shapes are depicted Figure 6.4 on page 114. Obviously, algorithms preferring circle-shaped compact clusters are not a good choice for datasets consisting of stripe-shaped clusters. As the nature of the cluster shape is not necessarily known *a priori* or can even be of mixed form, the clustering tool should be indifferent in preferring a certain shape.

Minimal user-specified input: The algorithm's behavior should depend on as few as possible parameters.

All partitional clustering algorithms discussed in this thesis require at least one parameter to be set in order to influence the behavior of the algorithm. A multitude of available parameters may cause problems in practice. For instance, tools with many parameters are prone to over-training. Generally, finding good parameters becomes increasingly difficult with a growing parameter-space, often leading to the use of a suboptimal parameter set for a certain problem.

Interpretable results: A clustering result should be intuitively interpretable with respect to the given similarity function and the used parameters.

Ideally, the algorithm guarantees certain properties of the reported clustering result in dependency of the used parameter set. E.g., the parameter k of the widely used K-means approach defines the number of clusters in the result. Given parameters with such clear influence on the result, the user can intuitively decide how to change the parameter set in order to receive a clustering better suited to the given problem. In cases where the algorithm cannot provide such an intuitive connection between the results and the parameters, the practitioner needs to have a deep understanding of how the given clustering process works in order to influence the result in the desired way.

Reproducibility of results: Running the algorithm several times on the same dataset should yield the same results.

This is given for all deterministic algorithms. If the result differs in each run, it is hard to find an optimal parameter set as the quality of the result is not solely dependent of the parameters and the given input.

Availability: The clustering algorithm should be available as a reference implementation and in source code, preferably free to download.

This eases the integration of the algorithm into pipelines of larger projects and allows for the adaption of the program for certain application cases. An important point for lowering the entry barrier, especially for non-expert users, is the availability of a web-service performing small cluster analyses without the need for any installation.

In consideration of the rapid growth of the available biological data mentioned in the motivation (Section 1.1 on page 13), we define two additional requirements and will extend TransClust in the framework of this thesis to support these:

Missing Values: The algorithm should be able to support similarity files with explicitly missing values (refer to Subsection 2.2.5 on page 32 for the definition of explicitly missing values).

Biological datasets are not only prone to noise but may also be incomplete. This incompleteness can be caused by many different reasons, e.g., a certain measurement was never performed or is not measurable (e.g., it would require the knock-out of a vital gene) or can be intentional in order to reduce the runtime for the similarity calculation. A clustering algorithm should allow for those explicitly missing values and still produce meaningful results.

Active Clustering: The algorithm actively decides which similarities need to be calculated and which can be disregarded.

This point can be regarded as an extension of the "Missing Values" criteria. An active-clustering approach iteratively decides, based on intermediate results, which proximity measures need to be calculated in order to improve the result the most. This reduces the cost in terms of computational time and space for the creation and storage of the similarity values. For a detailed description of active clustering, refer to Chapter 5 on page 85.

Even though clustering is a long standing problem in computer science and has obviously been tackled several times, there is no clustering tool fulfilling all criteria. Some of them miss features which can be implemented (e.g., integration of prior knowledge), or are intrinsically tied to the algorithmic approach (e.g., non-reproducibility of a random algorithm). Which features are present in which tool is summarized in Table 2.3.

Among the major challenges when performing a cluster analysis are the optimal selection of the clustering tool, and the correct tuning of the tool's parameters. These tasks require considerable knowledge from the researcher, especially with regard to the methodological diversity with which the different algorithms tackle the clustering problem. It is apparent that many details of an algorithm in question must be known before one can efficiently fine-tune the different parameters. That becomes even more apparent when the user has to specify parameters which have only indirect influence on the result and represent some internal implementation detail (e.g., the length of the tabu list of RNSC, see Subsection 2.4.8 on page 47). Only a few algorithms have parameters which have a provable influence on the results: K-means where k defines the number of clusters, and TransClust, where it is guaranteed that all clusters have an average intra-cluster similarity above the threshold t and respectively, an inter-cluster similarity below t [125]. CFinder has (among other parameters) the parameter k which asserts that all reported clusters are at least k-cliques.

2.5. Evaluation Frameworks

As seen in the previous section and also laid out in the motivation, the practitioner faces a plethora of different clustering algorithms. In order to decide for a certain

algorithm and use them in the most efficient and reliable way, a lot of prior knowledge regarding the algorithms is required. Even for an expert, a large cluster analysis, involving several datasets and clustering tools, remains a tedious and complicated task, as most clustering algorithms require different input formats and the parameters need to be trained for optimal results. Here, we give a short overview of existing frameworks supporting the researcher with conducting a cluster analysis.

2.5.1. Environment for Developing KDD-Applications Supported by Index-Structures (ELKI)

ELKI (Environment for Developing KDD-Applications Supported by Index-Structures) [2] is a software framework written in Java designed for knowledge discovery in databases. The authors mainly concentrate on the standardization and fair evaluation of so-called subspace clustering methods. These clustering methods are sometimes also called biclustering, especially in bioinformatics [69]. They normally reduce the problem space by looking for clusters in subspaces of the original space. The framework itself is programmed in Java and ships with a variety of implementations of clustering algorithms, especially subspace approaches. Among the general purpose algorithms are SLINK [106], k-means [75], EM-clustering [34], DBSCAN [45], Shared-Nearest-Neighbor-Clustering [44], OPTICS [8], and DeLiClu [1] (all algorithms referenced as in [2]). The framework is extensible but all implementations require implementation in Java for a seamless integration. The framework can handle several different input formats and provides interfaces for easily including more formats. The framework specializes mainly in the precise evaluation of clustering algorithms (in terms of runtime and memory usage) and does not support the user in performing a large-scale clustering study.

2.5.2. jClust

With jClust [95], Pavlopoulos *et al.* presented an approach to unify clustering approaches. The Framework is written in Java and includes some of the most widely used clustering approaches in bioinformatics: Affinity Propagation [49], K-means [75], Markov Clustering [42, 112] and Spectral Clustering [91]. They use one simple input and output format and provide a variety of different methods for post-processing the clustering results. The results can be visualized by means of an implementation of the Medusa visualization module [62]. The tool is not open source, and thus cannot be extended to use new clustering algorithms or different input formats.

2.5.3. The Konstanz Information Miner (KNIME)

The Konstanz Information Miner (KNIME) [20] is an environment for interactively assembling data pipelines. The framework is implemented as a plug-in for the widely used IDE (Integrated Development Environment) Eclipse[1] and thus bound to Java.

[1]The Eclipse Foundation. http://www.eclipse.org/

Again, the main focus of the framework is not clustering but the handling of data processing pipelines in general. The project is similar by means of functionality and concept to RapidMiner [82] (see Subsection 2.5.4). Therefore, parameter optimizations can only be included rudimentarily by including loops in the pipeline. The desktop version of KNIME is open source and provides an API for additional modules implemented by the user. The standard version ships only with an implementation of K-means and hierarchical clustering, but allows for the integration of WEKA [58] and RapidMiner modules. Due to the implementation as an Eclipse plug-in, KNIME provides a sophisticated user interface and provides a variety of different data visualizations.

2.5.4. RapidMiner

RapidMiner (formerly YALE - Yet Another Learning Environment) [82] follows a similar approach as the aforementioned KNIME. RapidMiner also represents tasks as nodes whose inputs can be connected to outputs of other nodes. It ships as a stand-alone program written in Java. The graphical interface also provides numerous data and cluster visualizations. As the name YALE already suggests, it is mainly tailored for machine learning processes and not specialized for large scale cluster analyses. RapidMiner provides its own scripting language and the source code with an extensive API for extending the framework. Thus, new methods as well as new proximity measures can be included. For finding optimal density parameters, RapidMiner provides basic parameter optimization. Furthermore, comparable to KNIME, it also includes the WEKA library and thus ships with all clustering methods provided by WEKA.

2.5.5. The Waikato Environment for Knowledge Analysis (WEKA)

The Waikato Environment for Knowledge Analysis (WEKA) [58] is probably one of the best known applications for statistical learning and data mining. The main foci of WEKA are typical machine learning tasks such as classification, support vector machines, decision tree learning and many others. In contrast, WEKA only rudimentarily supports clustering. WEKA ships only with a few standard clustering algorithms, including EM clustering, K-means, and hierarchical clustering. By means of the plug-in system, a few more clustering algorithms can be installed. Since version 3 appeared in 1999, WEKA is implemented in JAVA and available as an open source program [58]. Thus, WEKA can be extended either by plug-ins or by extending the source code itself. WEKA defined an own file format, called ARFF (Attribute Relation File Format) which is quite commonly used in machine learning tasks, but also supports several other file formats. WEKA also supports the user by means of a graphical user interface modeling data workflows, but lacks for methods of automated threshold probing. The GUI also provides a range of different dataset analysis tools and graphical representations. One of the standard text books for machine learning, “Data Mining: Practical Machine Learning Tools and Techniques” by

	ELKI	jClust	KNIME	RapidMiner	WEKA
Integration of Existing Tools	(✔)	(✔)	(✔)	(✔)	(✔)
Data analysis	(✔)	✗	✔	✔	✔
Cluster Evaluation	✗	✗	✗	✗	✗
Parameter Optimization	✗	✗	(✔)	(✔)	✗
Extensibility	✔	✗	✔	✔	✔
Runtime Efficiency	✗	✗	(✔)	✗	✔
Availability	(✔)	(✔)	(✔)	(✔)	(✔)

Table 2.4.: Overview of features defined in the text which are fulfilled by the clustering tools. For criterion "Integration of existing tools" we have ranked all tools only as partly fulfilled as no tool provides a significant amount of the state of the art biological clustering tools but only ship with older and simpler methods like k-means or hierarchical clustering. The same applies for the criterion "Availability": All tools can freely be downloaded but no website provides an overview of the existing clustering tools and their performance which is especially useful for non-experts.

Ian Witten, Eibe Frank, and Mark Hall [119], is built around WEKA. This arguably helped WEKA to become the most important tool for machine learning.

2.5.6. Summary

The previous subsections introduced mainly machine-learning frameworks. These frameworks can be used to perform automated cluster analyses, but most of them have not been designed for the specific problems that arise during a cluster analysis. It is not possible to define one objective measure ranking the tools according to the capability conducting a cluster analysis. In the course of this thesis, we will develop the cluster evaluation framework ClustEval. Thus we will describe the requirements we have defined for the development of ClustEval:

Integration of existing tools: The ability of the framework to support most of the commonly used clustering algorithms in bioinformatics.

This is a crucial requirement for such a framework to get accepted in the clustering community. This allows the user to simultaneously apply one dataset to multiple algorithms. Therefore, the platform must support different input formats and data preprocessors in order to serve every clustering tool with the correct input.

Data analysis: The ability of the framework to calculate and present common statistics for a given dataset.

One major task before conducting a cluster analysis is the exploration of the dataset at hand. Here, the framework should assist the user with a wide variety of different measures providing a good overview of the properties of the dataset.

Cluster Evaluation: The ability of the framework to assess the quality of the clustering results by means of the most common internal and external quality measures.

This includes the calculation of different internal and external quality measures. Refer to Section 2.3 on page 32 for on overview of commonly used quality measures.

Parameter Optimization: The capability to detect the best parameter set for a clustering tool maximizing a specified quality measure for a given dataset.

Finding a good parameter set is one of the most time-consuming tasks when performing a cluster analysis. Here, the framework should support methods for automated parameter testing. This also increases the comparability between different clustering tools, as both were tested with parameter sets derived from a standardized optimization procedure.

Extensibility: The availability of mechanisms allowing for extensions of all parts of the framework by the user.

This criterion describes the ability of the framework to include new clustering algorithms, datasets, data formats, quality measures, etc. Additionally, the ability of the framework to invoke binaries is very important to support closed source programs.

Runtime Efficiency: The ability of the framework to efficiently utilize all available resources.

As already mentioned in the motivation, clustering is a computational intensive task. Thus, the framework should be able to use the available computing resources as efficient as possible, e.g., parallel execution on multi-core computers.

Availability: The framework should be available as an open-source project.

Furthermore, the framework should be accompanied with a website presenting the performance of the different tools on different datasets. This supports especially non-expert users to judge the quality of different clustering tools and find the best suited tool for a given problem.

Table 2.4 summarizes the existing tools and their features. No existing tool fulfills all the requirements which we described here, mainly because none of the tools was specifically designed for cluster analyses. All tools lack a convenient way of integrating binary executables of clustering tools and thus hinder the integration of tools with no open-source reference implementation. Automated parameter probing is only partly fulfilled by two frameworks, mainly due to the lack of the most prominent cluster quality measures.

2.6. Datasets

To finish up the introduction, we briefly describe two dataset we will use throughout the entire thesis. The Brown *et al.* [23] dataset is widely used for assessing the performance of clustering algorithms. For this dataset, a gold standard is available. The description is based on our previous publication [101]. The Actinobacteria dataset emerged during a project published in [100]. This dataset is significantly larger as the Brown *et al.* dataset but does not have a gold standard.

2.6.1. Brown *et al.* Dataset

The gold standard of Brown *et al.* [23], published in 2006, is a widely used dataset for clustering approaches in bioinformatics. The dataset comprises of five enzyme superfamilies (amidohydrolase, crotonase, enolase, haloacid dehalogenase, and vicinal oxygen chelate) with different levels of sequence diversity. On the one hand, the enolase and crotonase superfamilies contain a very homogeneous set of sequences, i.e., high sequence similarities thus are expected to be easily clustered. The other extreme are the haloacid dehalogenase and parts of the amidohydrolase, which include a very divers set of sequences with a comparably high number of outliers. Therefore, this set of protein sequences serves as a suitable evaluation dataset for clustering tools. The five superfamilies consist of 4,887 proteins which are further divided into 91 families. Each of the amino-acid sequences is either annotated to a gold or a silver standard family. Gold standard families only contain sequences with experimentally determined functions, while silver standard families are less restrictive. As done in previous studies, when we compared Transitivity Clustering to other approaches [122], we only used the 866 sequences that are assigned to a gold standard family.

2.6.2. Actinobacteria Dataset

One common task in bioinformatics is the protein homology detection. In the framework of the aforementioned paper "Density parameter estimation for finding clusters of homologous proteins - tracing actinobacterial pathogenicity lifestyles" [100] we compiled a dataset consisting of the protein sequences of 89 different actinobacteria. The phylum actinobacteria is one of the biggest clades of bacteria. Their members show a high diversity throughout different lifestyles and can cope with a variety of different habitats [81]. Many of these bacteria are important for biotechnological production processes, as well as human and animal medicine [113]. In this dataset, we focus on selected species of the following so-called CMNR group: *corynebacteria*, *mycobacteria*, *nocardia* and *rhodococcus*. Our main focus of attention is *Corynebacterium pseudotuberculosis*. It causes caseous lymphadenitis in animals [118], with dramatic effects on livestock all over the world. All CMNR organisms selected for this dataset share common properties with impact on the design of effective vaccinations; they all share a common cell wall organization [37], for instance. For drug target detection, accurate homology information about the protein space in this group is important,

e.g., for reducing drug side effects and negative effects on the other microorganisms that are part of the host's microbiome.

We obtained the protein sequences in FASTA format from NCBI [86] for the 89 sequenced and annotated actinobacteria of the CMNR group. See Table C.1 on page 211 in the appendix for a list of all species and a classification into the four pathogencity classes. We also report the associated disease where available. Our dataset comprises 344,421 proteins of 89 species: 27 corynebacteria, 55 mycobacteria, 6 rhodococcus and 1 nocardia. We use this dataset throughout the entire thesis as an example for a large dataset posing a computational challenge for clustering tools. In some chapters, we use a reduced dataset (the "Coryne dataset") consisting only of the 27 corynebacteria of the entire dataset.

3. Handling of Large Scale Similarity Files

Beginning with this first method section, we concentrate on extending transitivity clustering and its implementation TransClust. Some of the presented ideas could also be implemented for other clustering tools, but we limit ourselves to TransClust for the following reasons:

- TransClust already supports most of the desired features of a state of the art clustering tool (compare to Subsection 2.4.11 on page 53).
- TransClust has shown on several occasions to rank amongst the best clustering tools commonly used in bioinformatics (e.g., refer to [121, 125]).
- TransClust is based on the WTGPP (refer to Subsection 2.4.10 on page 49) for which is proven that the average similarities between objects of one cluster are above the threshold whereas the similarities of objects between clusters are below the threshold on average [125]. We will exploit this property for extending TransClust in Chapter 4 on page 77 and 5 on page 85.

Objectives of this Chapter

- Enable the decomposition of large similarity files in connected components on standard desktop computers.
- Development of a cost-matrix creator efficiently utilizing background storage to account for limited main memory.

3.1. Overview and Problem Statement

3.1.1. Problem Statement & Introduction

As mentioned in Subsection 2.4.10 on page 49, each connected component of a given input can be treated as an independent problem instance. This decomposition reduces the problem size for solving the WTGPP as in most practical applications the different

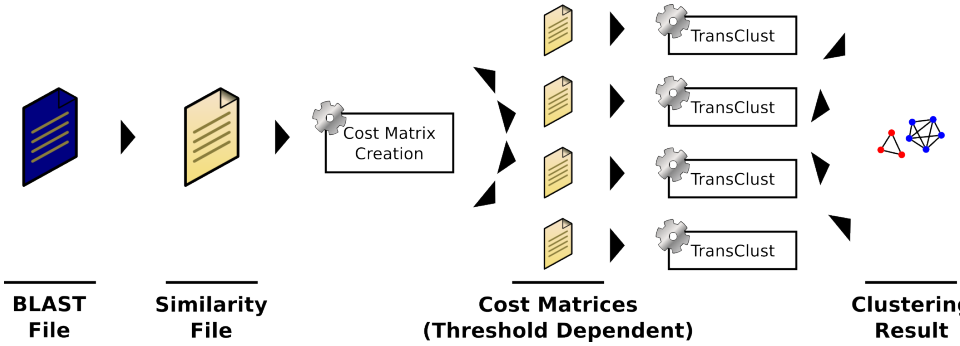


Figure 3.1.: This figure depicts a typical TransClust workflow. The input similarity file is separated into connected components and written into cost-matrices. These cost-matrices are treated as independent problem instances. The clusters resulting from each of these connected components are combined to the clustering result reported by TransClust.

connected components are significantly smaller than the original input graph. Given a similarity file as input, a typical TransClust workflow consists of the following steps (also depicted in Figure 3.1):

1. Identification of the connected components
2. Creation of the cost-matrix files
3. Individual clustering of each cost-matrix file
4. Combination of the individual clustering results to the overall clustering result

A cost-matrix represents a connected component. In contrast to the similarity file, such a cost-matrix stores the costs for each edge modification and is therefore threshold dependent. For each pair of objects $u, v \in V$, the cost-matrix file stores the edit costs $c(u, v) = |t - s(u, v)|$.

During the entire clustering process, the decomposition of the similarity file into connected components is the only time where the complete input (i.e., the similarity file) needs to be processed as a whole. Finding these connected components is usually done with a depth- or breadth-first search (DFS or BFS respectively) on the given input graph. Implementing such a DFS efficiently, the entire graph with all its edges must be stored in the main memory of the computer. Storing the edges can pose problems as their number grows quadratically with the number of nodes: $\binom{|V|}{2}$. Depending on the input size, this task can already be infeasible on modest computing hardware. For instance, a homology detection involving 50 organisms with 2000 genes each results in ~ 5 billion similarities. When stored as `double` (8 bytes following IEEE 754), only the similarities require ~ 37 GB to be held in main memory. In case of protein sequences and BLAST as similarity function, this problem can still be solved with normal computers, as most similarities are zero (i.e., below the BLAST E-value cut-off) and thus do not need to be stored explicitly. Whenever a different

similarity function without this favorable property is used, it is quite apparent that even rather small studies will hit the boundaries of today's computers.

In order to enable users with restricted access to computers with large main memory the decomposition of the similarity matrix into connected components, we developed an extended cost-matrix creator (CMC) by utilizing *external* algorithms. An *external* algorithm keeps most of the information on external memory such as the hard disk, and holds only the currently relevant part of the task in main memory. The main bottleneck here is obviously the I/O performance of the computer in question. Hence the application of less efficient algorithms (in terms of runtime behavior) which reduce the need for I/O accesses can be beneficial. In the following section we will discuss the used algorithms and evaluate runtime and memory consumption in comparison to the currently available CMC included in TransClust.

In this work, we always consider the use of a normal general-purpose computer with a hierarchical memory setup. We distinguish between *external* storage, the background storage devices like hard drives, and *internal* storage, the main memory. We do not consider the different levels of cache hierarchy as we present a general approach to the problem implemented in Java. The access times of the main memory and the background memory differ by a factor of approximately a million ($\sim 10^{-9}$ seconds compared to $\sim 10^{-3}$ seconds) [115], thus the usage of the background storage should be limited to a minimum. In the following, we call an algorithm *internal* when all data required for the execution fit in the main memory, and *external* when the background storage is used in order to cope with the data.

3.1.2. Requirements for a Memory Efficient Cost-Matrix Creator

A common type of input for clustering algorithms used in bioinformatics is a similarity file. Such a file already contains all pairs of nodes and specifies the proximity between them and the calculation of those similarities is not performed by the clustering algorithm itself (in contrast to algorithms which take as input the geometrical coordinates of the objects and apply, e.g., metric-based distances). Usually, specialized programs are utilized to calculate the similarities, e.g., in case of protein sequences programs like BLAST.

Furthermore, in most publications and analyses of clustering algorithms the existence of such a similarity file is simply assumed. Nevertheless, this neglects the important and time-consuming construction of such a similarity file which can create a hurdle for non-expert practitioners. This is the reason that we do not only support the usage of an already existing similarity file, but also the direct usage of BLAST and FASTA files, as this is a common application in bioinformatics. In the method section, we first describe how CMC transforms such a BLAST and FASTA file into a regular similarity file; then we proceed with the decomposition of a similarity file into cost-matrices.

To summarize, in this part of the thesis, we extend the currently built-in cost-matrix creation functionality of TransClust to support the following main features:

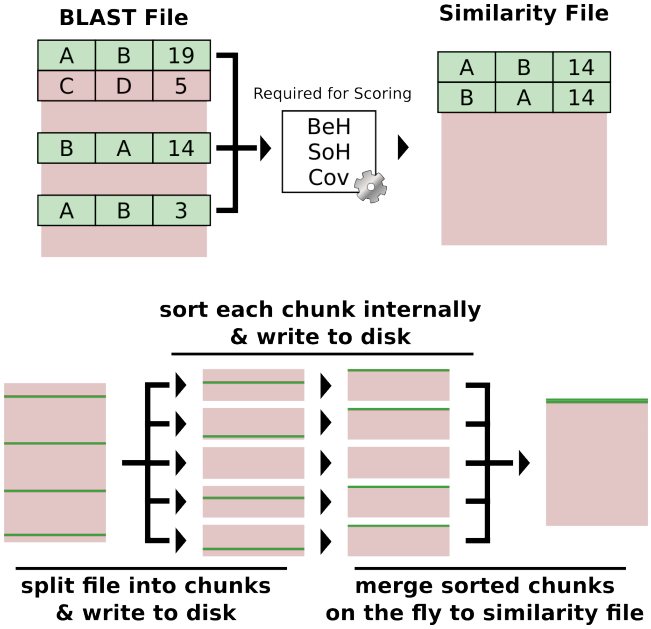


Figure 3.2.: The conversion of a BLAST file to a similarity file. The upper part of the figure demonstrates that the entries required for scoring the protein pair A, B can be spread over the entire BLAST file (required entries are indicated with green and all other entries are represented as the light red). The lower part depicts the process of the conversion of a BLAST file into a sorted similarity file (the green bars represent the relevant entries for a protein-pair). First, the BLAST file is split into chunks which can then be separately sorted using an internal algorithm. With these sorted chunks, the protein pairs can easily be scored and written into a sorted similarity file on the fly.

- Direct handling of BLAST and FASTA files with different cost models (refer to Subsection 2.2.3 on page 29)
- No sorting of the input files is required
- Large files must be efficiently processed even on computers with limited main memory

3.2. Methods for Memory-Aware Similarity Processing

3.2.1. BLAST and FASTA to Similarity File

As all HSPs of a pair of proteins in a BLAST file can be distributed over the entire file, it is not possible to convert an arbitrary BLAST file into a similarity file on the fly without storing all entries in the main memory (compare to the top-part of Figure 3.2) as all HSPs for a protein pair are required for the calculation of the similarity value. To be able to process such a BLAST file on the fly with limited main memory, all HSPs for a certain protein pair need to appear subsequently in the file, i.e., the

BLAST file needs to be sorted ("sorted" means that all HSPs of one pair are grouped together, otherwise no specific order of the protein IDs is required). With that, the sorted BLAST file can easily be read in small chunks and the similarities can be calculated and written into the similarity file. The sorting itself is handled by our CostMatrixCreator as we do not want to impose the task of sorting to the user.

The sorting must be accomplished with an external algorithm. All such algorithms consists basically of two parts, an *external* and an *internal* part. Whilst the internal part sorts chunks of data which fit into the main memory, the external part sorts large portions of the total file [114] which do not fit in the main memory. External sorting is a long standing problem and very well understood [115]. As we aim for usage on modest computing hardware, we can neglect load balancing problems caused by using several disks in parallel. Basically, two main methods have been established, sorting by *distribution* (equivalent to the internal bucket sort) and sorting by *merging* (equivalent to the internal merge sort) [115]. In the one-disk/one-CPU setting which we are considering, both methods have an almost identical performance.

For a setting as presented here, external sorting by merging is the favored method as sorting by distribution would require the development of a hash-function distributing pairs of proteins in equal chunks in order to yield optimal performance. Let us assume a BLAST file with a total of N entries and an internal memory which can hold up to Q entries. Sorting by merging consists of the following steps (compare to the lower-part of Figure 3.2):

1. Read Q entries of the original file and sort them with an internal sorting algorithm.
2. Write this sorted chunk to the disk.
3. Repeat until the entire original file has been processed, i.e., $\left\lceil \frac{N}{Q} \right\rceil$ chunks are written.
4. Open all sorted chunks and subsequently read the entries until all HSPs of a protein pair are collected. Calculate the final similarity and write the according line into the similarity file.

For the internal sorting, there are many possible algorithms available. Only algorithms operating *in-situ*[1] should be chosen, otherwise the number of entries Q which can be sorted in the main memory would be reduced. In our implementation, we facilitated heapsort as it has an optimal worst case performance for sorting of $\mathcal{O}(n \cdot \log n)$ and does not require recursive function calls, thus keeping the required auxiliary space complexity at $\mathcal{O}(1)$.

3.2.2. Similarity File to Cost Matrices

In this subsection, we assume to have a similarity file which exceeds the capacity of the internal memory as input. This file must be split into connected components which

[1] An algorithm is called in-situ if the execution of the algorithm only requires a constant amount of memory besides the input, i.e., the memory overhead is $\mathcal{O}(1)$ for every input.

then may be clustered. In order to ease the construction of the connected components, the similarity file is sorted. Let $S = (s_{i,j}) \in \mathbb{R}^{n \times n}$ be the similarity matrix with n being the number of nodes and $s_{i,j}$ the similarity $s(u_i, u_j)$ with $u_i, u_j \in V$. The similarity file is sorted in such a way that the entries appear in the following order:

$$s_{1,2}, \ldots, s_{1,n}, s_{2,3}, \ldots, s_{2,n}, \ldots, s_{k,k+1}, \ldots, s_{k,n}, \ldots, s_{n-1,n}.$$

This step is unnecessary when processing a similarity file which was created from the BLAST file described in the previous subsection. The sorting of an arbitrary similarity file is performed with the algorithm also presented in the previous Subsection 3.2.1.

In order to decompose the file into connected components, mechanisms like DFS or BFS are infeasible for this task, as the storage of the edges and their weights exceed the internal memory. Thus, we are facilitating a different data structure, namely a disjoint-set or union-find data structure with path compression. Such a data structure can be implemented as a forest of rooted trees, where each tree represents one distinct set and supports two operations:

Find(u) This operation traverses the tree of u upwards and reports the root v of that tree as the representative of the set u belongs to. Two objects are in one set whenever they report the same root v.

Union(u, v) With this operation, the two sets of u and v are united. This is accomplished by connecting the root of one tree to the root of the other tree.

This can be implemented using an array and the value of each entry points to the entry of the parent. Note, that the union-find data structure is also feasible to be utilized in an in-memory approach. In order to increase the efficiency of the find operations, path-compression is used. Whenever the representative of an object is searched, all nodes on the way up to the root are connected to the root. Hence, after the first find on u, the next find is accomplished in $\mathcal{O}(1)$ time. In total, that reduces the amortized costs for both operations to $\mathcal{O}(\alpha(n))$ with $\alpha(n)$ being the inverse Ackermann function [110], whereas the space requirement is $\mathcal{O}(n)$. The entire data structure is built on the fly and requires only one read of the similarity file.

In the case that the entire similarity matrix fits into the main memory, the creation of the individual cost-matrix files is straightforward. For each connected component, all similarities are looked up and in case of an implicitly missing value, replaced by the fall-back value and written to the output file. This is no longer possible if the similarities are stored on external memory, as in our case. Again, in order to limit the I/O load, a computationally less efficient approach is chosen which requires only one additional reading of the file. As stated above, the similarities are in the same order as a line-wise traversal through the entire "virtual" similarity matrix. We use the term "virtual" here because the entire similarity matrix is never explicitly stored due to the many implicitly missing values. The algorithm iterates over all pairs of objects u, v in the same order as the similarity file. For each pair, it is checked whether

		Connected Components			
Organisms	**Sequences**	**Total**	**Non-Transitive**	**Ø**	**File Size [MB]**
30	116,361	36,495	2,191	3.19	342
40	158,200	41,493	2,350	3.81	660
50	193,432	49,951	2,619	3.87	945
60	243,641	62,055	3,392	3.92	1,438
70	271,626	67,309	3,534	4.04	1,721
80	305,166	71,674	3,616	4.26	2,244
89	*344,421*	*78,734*	*3,969*	*4.37*	*2,845*

Table 3.1.: The datasets used for the evaluation. These are all sub-datasets of the original Actino dataset (bottom line in italics). The column "Connected Components" contains the following information based on a threshold of 48: "Total" - number of connected components (including transitive components and singletons); "Non-Transitive" - number of non-transitive components; "Ø" - average size of a connected component.

the two objects u and v are in the same connected component. If so, there are two possibilities:

1. The next entry of the similarity file corresponds to the similarity $s(u, v)$ of u and v. In this case, the costs (normally $c(u, v) = |s(u, v) - t|$) are calculated from the similarity and written in the cost-matrix file. Afterwards, the next entry of the similarity file is read.

2. The next entry in the similarity file is not the similarity $s(u, v)$ of u and v. Hence, this similarity is an indirectly missing similarity and is replaced with the user defined fall-back value.

The drawback of this approach is that many file-handlers need to be kept open or repeatedly opened and closed (depending on the limitation of the operating system and the available main memory) as the similarities are not sorted by their membership to connected components. On the other hand, this approach requires only one read of the similarity file and thus minimizes the I/O operations.

3.3. Results & Discussion

3.3.1. Runtime Analysis and Memory Requirements

First, we analyze the runtime behavior of the memory-aware CMC. We hypothesized that the main factor determining the runtime is the I/O load, which should lead to a linear growth of the runtime with the growing size of the input files. Nevertheless, the writing of the cost-matrices must be achieved by using an algorithm with a quadratic runtime; thus, the overall scaling can also be driven by that part of the algorithm. For the analysis, we used the Actino dataset (refer to Subsection 2.6.2) containing

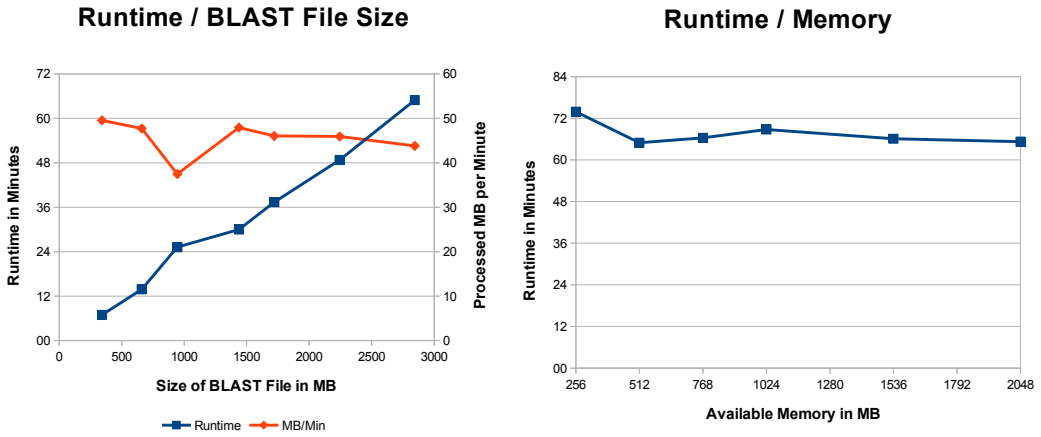


Figure 3.3.: The left figure shows the runtime behavior of CMC depending on the size of the BLAST input file. CMC was set up to use only 512 MB of heap space and shows a linear scaling. The red line further indicates the linear growth as the rate of processed BLAST file per minute remains constant. The right part of the figure depicts the runtime behavior with varying heap space. Here, only the complete Actino dataset was used. As the main bottleneck is the I/O load which remains constant, the overall runtime also remains constant.

the proteins of 89 bacteria with a BLAST file size of 2,800 MB. We also created subsamples of this dataset including less organisms. All datasets are summarized in Table 2.6.2 on page 62. All runtime tests were executed on a 24 core XEON server with 100 GB of RAM running a 64 bit Linux system. Note that the given times are reflecting the actual runtime and can vary slightly.

The results indicate that in fact, the new CMC scales linearly with the size of the BLAST input file (see Figure 3.3). Even though the output of the cost-matrices takes by far most of the time and is a quadratic algorithm, the runtime is determined by the I/O operations. Figure 3.4 depicts the memory usage as well as the I/O load over the execution time using the entire Actino dataset with threshold 48 which corresponds to an BLAST E-value of 10^{-48}. This threshold was selected as this results in the best clustering for protein homology detection and thus resembles the most relevant scenario (refer to Section 7.2 on page 136 for further details on the threshold). The reason why the algorithm scales linearly even though most of the runtime is spent in a quadratic algorithm is the usage of the file handlers. For example, for the entire Actino dataset, almost four thousand files, need to be written concurrently. As the number of parallel file handlers is limited (both by the operating system as well as memory wise as every file handler produces overload, e.g., for buffering), the CMC is required to open currently required files while closing others repeatedly. The overhead of these operations is the main factor for the runtime which also reflects the quite low writing rate during the cost-matrix creation. Thus, even though the algorithm scales quadratically, the runtime is determined by the number of files which in turn is determined by the number of connected components. Table 3.1 clearly shows that

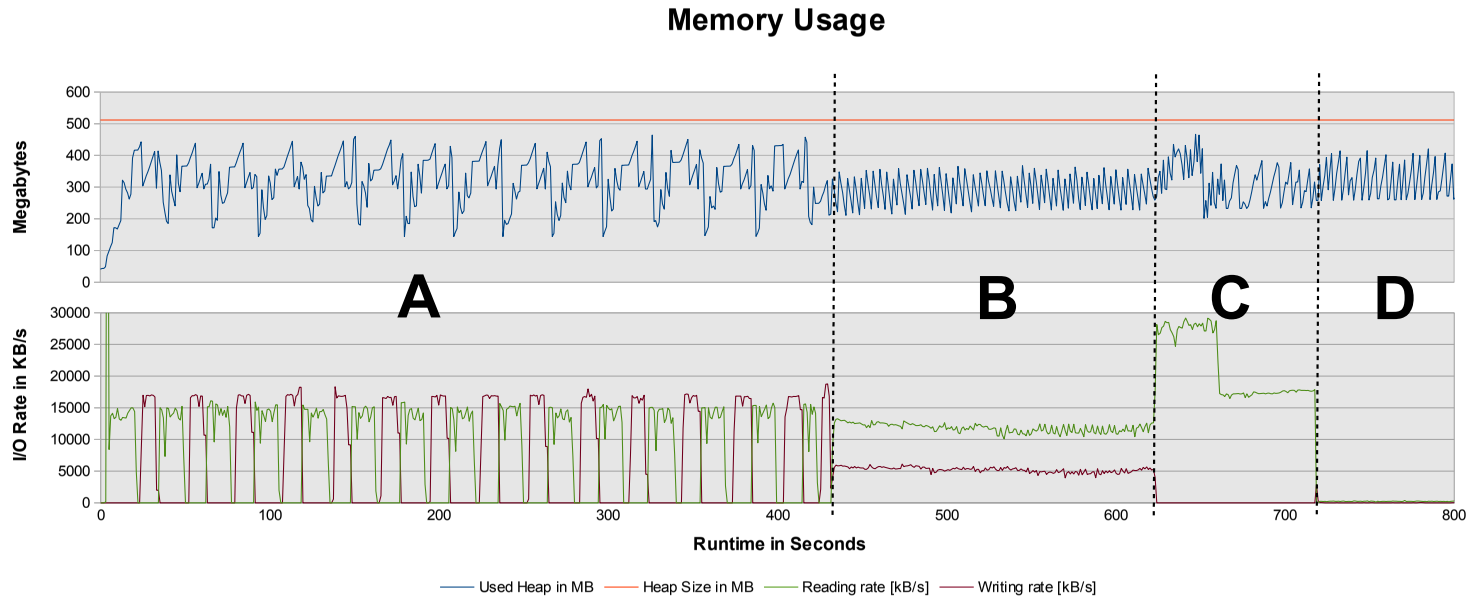


Figure 3.4.: Memory and I/O utilization of CMC while processing the entire Actino dataset with 512 MB of heap space. The picture is separated into four stages of the cost-matrix creation process: **(A)** split and sort the BLAST file into chunks (can be seen on the alternating read/write load, the small gap between reading and writing is due to the internal sorting; used heap space drops after each write process); **(B)** reading of the sorted chunks and merging into one similarity file (concurrent reading and writing; due to the redundant HSPs of some protein-pairs, the amount of written data is smaller than the amount of the read data); **(C)** reading the similarity file and connected component detection by means of the Union-Find data structure; **(D)** writing of the actual cost-matrix files (low I/O utilization due to the overload of the repeated opening and closing of file handlers). To increase readability, the graph is truncated after 800 seconds as the behavior of the graph does not change while writing the cost-matrices. The entire run lasted for about 3800 seconds. Furthermore, note that due to caching issues, the profiler reports an exceedingly high read-rate for the first 10 seconds which is simply truncated at 50 MB/s in order to increase readability of the plot.

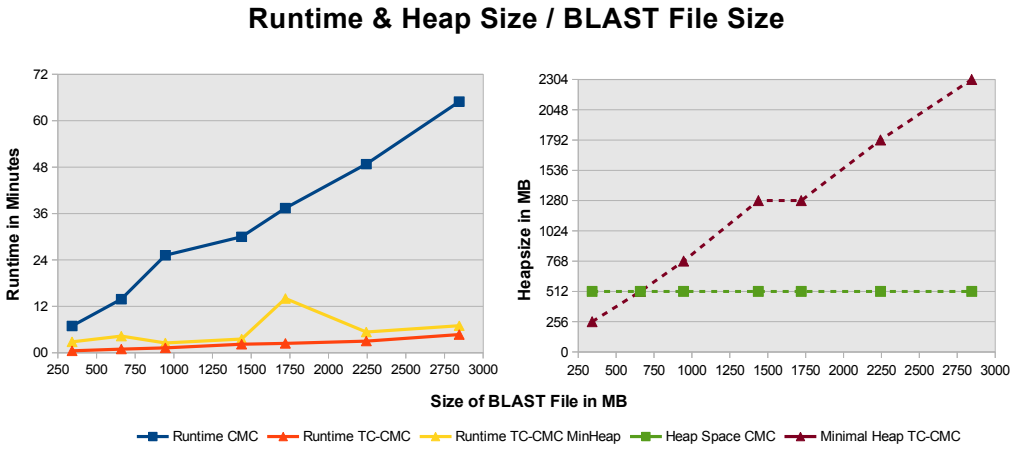


Figure 3.5.: the left side depicts the runtime of the new CMC (named "CMC") against the built-in CMC of TransClust (named "TC-CMC"). Both programs show a linear relationship between runtime and input file size. The line "Runtime TC-CMC" depicts the runtime of TC-CMC without any memory restrictions, whereas "Runtime TC-CMC MinHeap" shows the runtime when TC-CMC was limited to the smallest heap-size possible (slow-down due to increased garbage collection activity). The right figure represents the minimal memory requirement of TC-CMC (dark red line) for each dataset compared to the new CMC (green line).

the number of connected components grows even sub-linearly (which can also be seen on the fact that the average size of the connected components grows from 3.19 for the smallest dataset to 4.37 for the entire dataset) and thus compensates for the quadratic algorithm.

As the influences discussed above on the runtime are determined mainly by external circumstances, the amount of available physical memory should not affect the runtime significantly. In fact, as shown at the right side of Figure 3.3, the runtime of CMC remains constant when varying the available heap space from 256 MB to 2 GB.

3.3.2. Comparison to the Original Cost-Matrix Creator

After analyzing the runtime and memory behavior of the novel CMC, we want to compare the performance to the built-in version of CMC in TransClust. In the remainder of this section, TC-CMC refers to the built-in version of the cost-matrix creator whereas CMC refers to the new approach. As expected, the original version is significantly faster than the novel approach. The main reason for this is that the legacy version can write the cost-matrices far more efficiently than the new memory aware version. Figure 3.5 depicts the runtime for both versions, again using the datasets summarized in Table 3.1.

The minimal memory requirement for TC-CMC (probed in 256 MB steps) grows linearly with the size of the BLAST file. Due to the increased activity of the garbage collection, the runtime of TC-CMC more than doubles in some cases (depending on

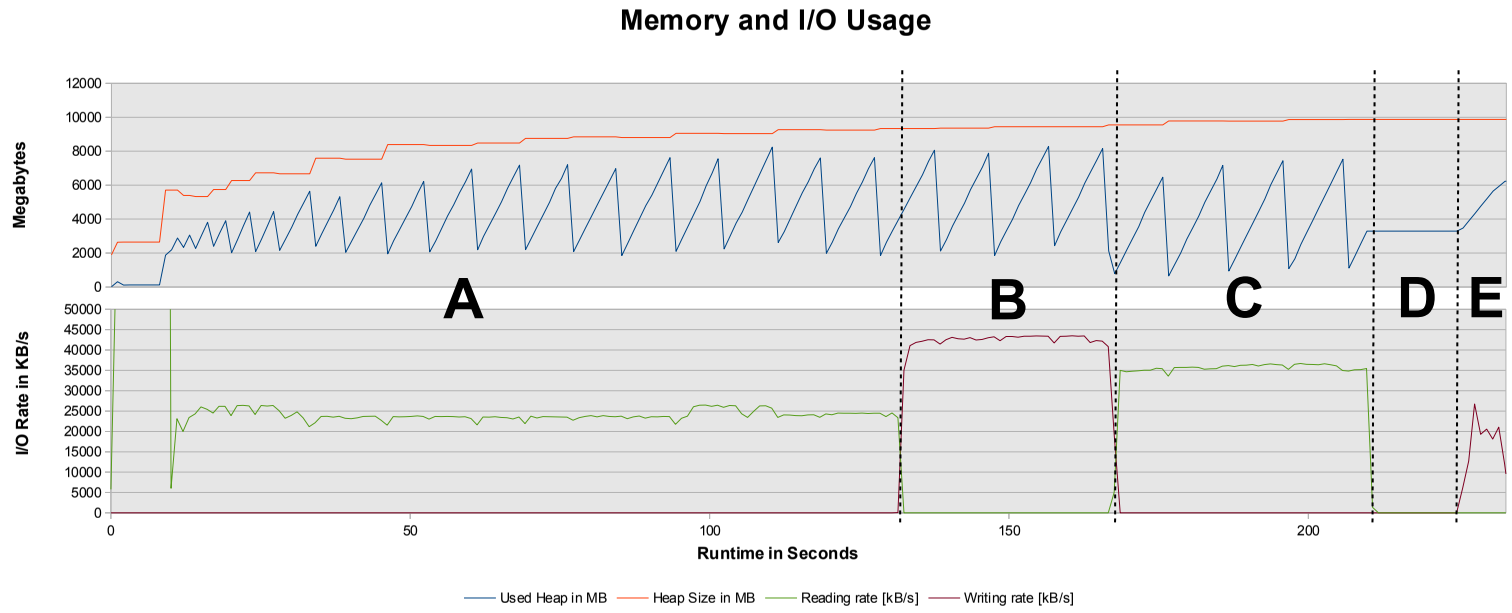


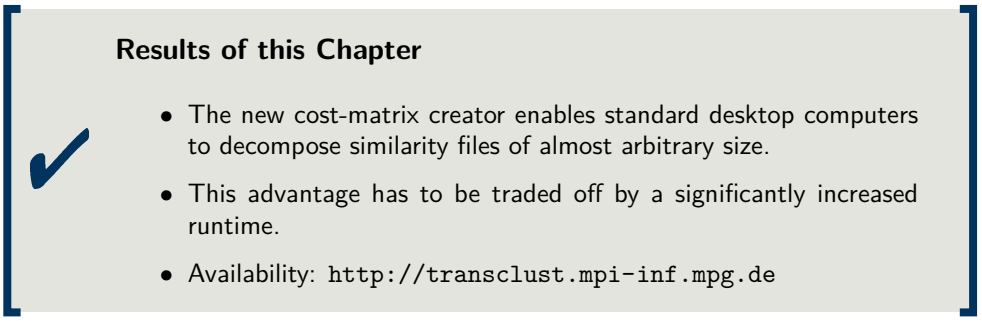
Figure 3.6.: Memory and I/O utilization of the TC-CMC while processing the entire Actino dataset without memory restriction. The picture is separated into five stages of the cost-matrix creation process of TC-CMC: **(A)** reading of the BLAST file; **(B)** calculation of the similarities and output to the similarity file; **(C)** all internal data structures for reading and processing the BLAST file are deleted (also seen as a large drop in the heap utilization), later reading of the similarity file with more efficient data structures for that purpose; **(D)** detecting of connected components; **(E)** output of the cost-matrices. Note that due to caching issues, the profiler reports an exceedingly high read-rate for the first 10 seconds which is simply truncated at 50 MB/s in order to increase readability of the plot.

how closely the 256 MB steps are to an absolute minimum). Nevertheless, as a rule of thumb, TC-CMC consumes about as much memory as the size of the BLAST input file. That limits the application of TransClust on modern personal computers to around 8 GB (approximately less than 600,000 sequences), which is crucial as many biologists do not have access to a larger shared memory server infrastructure. In comparison, the minimal memory requirement of CMC scales with the number of nodes, as they need to be constantly held in the main memory which grow at a much slower pace. In the used datasets, tripling the number of sequences lead to an eightfold similarity file size.

3.3.3. Conclusion & Discussion

The approach of decomposing similarities, or rather, BLAST files, into connected components and their associated cost-matrices now enables handling very large similarity files exceeding the main memory size. The analysis shows that the runtime scales linearly with the size of the input file as the main bottleneck is the overload for the concurrently required file-handler, which are directly connected to the number of connected components in the dataset.

On the other hand, the inefficient output of the cost-matrices has a large impact on the overall runtime, increasing it by the factor of 10 to 20. Even though memory requirements no longer pose a problem, this advantage is traded off by a much worse runtime. To conclude, the decomposing of large similarity files still poses problems either in terms of memory usage or in terms of runtime. More sophisticated strategies to limit the overload by the file handlers (e.g., keeping a pool of frequently used output files open) may soften that issue but cannot solve the underlying general problem of the actual size of the similarity files.



In the next chapters, we will discuss approaches on how to avoid large similarity files by drastically reducing the amount of required similarities for clustering in the first place.

4. Clustering with Missing Values

This chapter is based on the publication [101]:

Richard Röttger, Christoph Kreutzer, Thuy Duong Vu, Tobias Wittkop, Jan Baumbach. **Online Transitivity Clustering of Biological Data with Missing Values.** *German Conference on Bioinformatics 2012*, 57-68, ISBN: 978-3-939897-44-6, DOI: 10.4230/OASIcs.GCB.2012.57

Objectives of this Chapter

- The calculation of the similarity file is computational expensive and limits the applicability of clustering algorithms.
- Development of a method for exploiting missing values in datasets; thus reducing the number of required similarities.

4.1. The Moving Bottleneck

In the previous chapter, the handling of large similarity files, i.e., the decomposition into connected components, was discussed. Although the new algorithm allows for the handling of very large similarity files, it increases the runtime. Thus, the size of the similarity files still poses problems, albeit of a lesser nature, during the clustering analysis.

In addition to the size of such a similarity file, a widely neglected issue for most clustering approaches is the computation of the similarities in the first place. Especially when using complex similarity functions, e.g., protein interaction predictions or protein structure predictions, the actual calculation of the similarities between the objects consumes most of the time [29, 67, 61].

An indication how to widen the bottleneck arises from the definition of a clustering itself: we are looking for densely connected components sharing common traits. Hence, most similarities within one cluster are redundant in that they further confirm the similarities between the objects of a cluster (refer to the end of Subsection 4.2.1

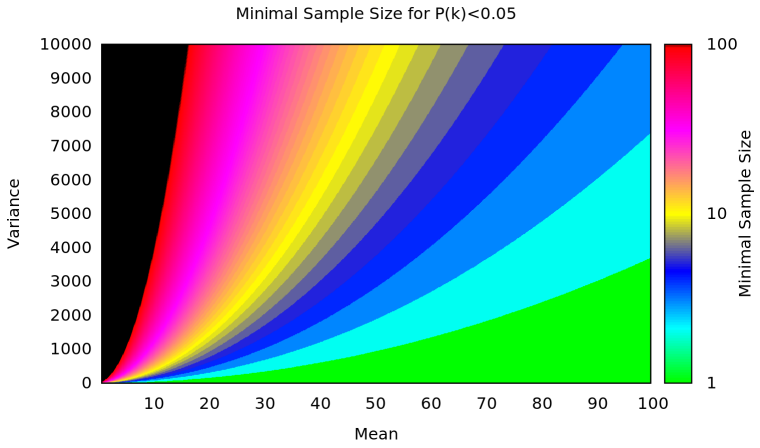


Figure 4.1.: This Figure depicts the minimal sample size k' for which $P(k') < 0.05$ in dependency of μ ("Mean") and σ^2 ("Variance"). The color black indicates $k > 100$. Note the logarithmic scale of the color scheme.

for a formal explanation). The similarities between clusters also tend to be very similar, namely low (as discussed in Subsection 2.2.5 on page 32). These observations suggest that a large share of the similarities can be excluded from the calculation in the first place without deteriorating the clustering result. Exploiting this fact by the utilization of missing values in order to limit the resources spent on the calculation of the similarities and as a consequence reduce the size of the similarity file appears to be a promising approach to tackle this problem.

In a previous study [125], we showed that TransClust is robust with respect to noise in the dataset. That makes TransClust a suitable approach for the exploitation of missing values. In this chapter, we demonstrate how explicitly missing values are accounted for in TransClust. We evaluate the influence of missing values on the clustering by comparing clusterings with missing values against the results of clusterings using full information. Similar to the previous study, we use protein homology detection as an example application. In the next chapter we will then extend this method to a more sophisticated active clustering approach.

4.2. Integration of Missing Values into TransClust

4.2.1. Extension of the Weighted Transitive Graph Projection Problem (WTGPP)

The similarity matrices used so far only contained indirectly missing values (refer to Subsection 2.2.5 on page 32). Thus, the edges E can be split into two distinct sets E_K and E_I. The set E_K denotes the edges $uv \in E_K$ with known similarities $s(u, v) \in \mathbb{R}$. The set E_I denotes those edges for which the similarity function did not return a value because their similarity is below the detection limit λ of the similarity function

(e.g., below the E-value cut-off of BLAST). In other words, an indirectly missing edge $uv \in E_I$ is not without any information but rather carries the information that $s(u,v) < \lambda$. This information was used to impute the indirectly missing values by a modified similarity function s':

$$s'(u,v) = \begin{cases} s(u,v) & \forall uv \in E_K \\ \lambda & \forall uv \in E_I \end{cases}$$

Depending on the used similarity function, λ either is set to a user-defined value (e.g., the E-value cut-off) or to the minimal measured similarity $\lambda = \min_{uv \in E_K} s(u,v)$.

In this chapter, we introduce explicitly missing values, i.e., similarities which were not calculated at all. Consequently, the set of edges E is split into three distinct sets E_K, E_I and E_E. For these explicitly missing values E_E we can not assume a boundary as for edges in E_I.

We adapt the weighted transitive graph projection problem (WTGPP) in order to handle explicitly missing values. As already stated in 2.4.10 on page 49, the total edit costs to transform a graph G_t (with t being the threshold) into a transitive graph G' is

$$\text{cost}(G_t \to G') = \underbrace{\sum_{uv \in E_t \setminus E'} |s(u,v) - t|}_{\text{deletion costs}} + \underbrace{\sum_{uv \in E' \setminus E_t} |s(u,v) - t|}_{\text{insertion costs}}.$$

In order to integrate a possibility to deal with missing edges in the graph G, we slightly adjust the underlying similarity function:

$$s^\star(u,v) = \begin{cases} s(u,v) & \forall uv \in E_K \\ \lambda & \forall uv \in E_I \\ t & \forall uv \in E_E \end{cases}$$

The similarity for edges $uv \in E_E$ is set to the user-given threshold t. As a result, the costs for adding/removing an edge $uv \in E_E$ is

$$\underbrace{|s^\star(u,v) - t|}_{\text{deletion cost}} = \underbrace{|s^\star(u,v) - t|}_{\text{insertion cost}} = |t - t| = 0.$$

As a consequence, the overall costs for transforming a graph G_t into the transitive G' is not affected by the missing values:

$$\text{cost}(E_E) = \sum_{uv \in E_E \cap (E_t \setminus E')} |s^\star(u,v) - t| = \sum_{uv \in E_E \cap (E' \setminus E_t)} |s^\star(u,v) - t| = 0.$$

Since the explicitly missing values have no information content, we adapted our approach such that they do not impact the clustering process. The remaining edges with existing similarity values have to account for the clustering result. We are

confident that this does not harm the quality of the clustering result significantly, since we assume that most of the information of the edges of a cluster is redundant.

This can be demonstrated on a simplified model. Let us assume the intra-cluster similarity is following a normal distribution $X_+ \sim \mathcal{N}(\mu, \sigma^2)$, the inter-cluster similarity a normal distribution $X_- \sim \mathcal{N}(-\mu, \sigma^2)$, and the threshold is set to $t = 0$. In [125], we have proven that the average intra-cluster similarity of each cluster is above the threshold. That means in turn, a connected component is falsely separated into several clusters if the missing values cause the average similarity of the present edge weights to drop below the threshold. First, we can give the probability that an edge sampled from $\mathcal{N}(\mu, \sigma^2)$ is below 0:

$$P[X_+ < 0] = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{-\mu}{\sqrt{2\sigma^2}}\right)\right).$$

When drawing k samples from the given normal distribution, the sampling distribution $\overline{X}_+$ itself follows

$$\overline{X}_+ \sim \mathcal{N}(\mu, \frac{\sigma^2}{k}).$$

The probability $P(k)$ that k samples are on average below the threshold can be calculated as

$$P(k) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{-\mu}{\sqrt{2\frac{\sigma^2}{k}}}\right)\right).$$

Because of the symmetry of the normal distribution, the identical argument holds for the probability that a sample of k similarities from the inter-cluster distribution X_- is above the threshold which would cause a false joining of two separated clusters.

Figure 4.1 on page 78 depicts the minimal sample size k' for which $P(k') < 0.05$ in dependency of μ and σ^2. The figure shows that only a small number of samples is required in order to ensure an average similarity above the threshold. In fact, whenever the variance $\sigma^2 < 3.69 \cdot \mu^2$ of the normal distribution, only 10 similarities are sufficient to ensure that $P(10) < 0.05$[1].

This model only estimates the probability that the remaining edges compensate the missing edges in terms of preserving an average similarity above the threshold for a given cluster. In Subsection 5.1.1 on page 86, we will discuss the likelihood that randomly selected edges form a connected component in the first place.

Even though biological datasets generally do not follow a normal distribution, the presented model can be used for a rough estimation as to how well a given dataset may be clustered with missing values[2]. For example, the intra-cluster distribution

[1] A quadratic relationship between μ and σ^2 ensures that $\operatorname{erf}\left(\frac{-\mu}{\sqrt{2\frac{\sigma^2}{k}}}\right)$ remains constant. The factor of 3.69 was determined numerically.

[2] According to the central limit theorem, the sample-mean distribution $\overline{X}$ will approximate a normal distribution if the sample size is sufficiently large enough, even though X itself does not follow a normal distribution.

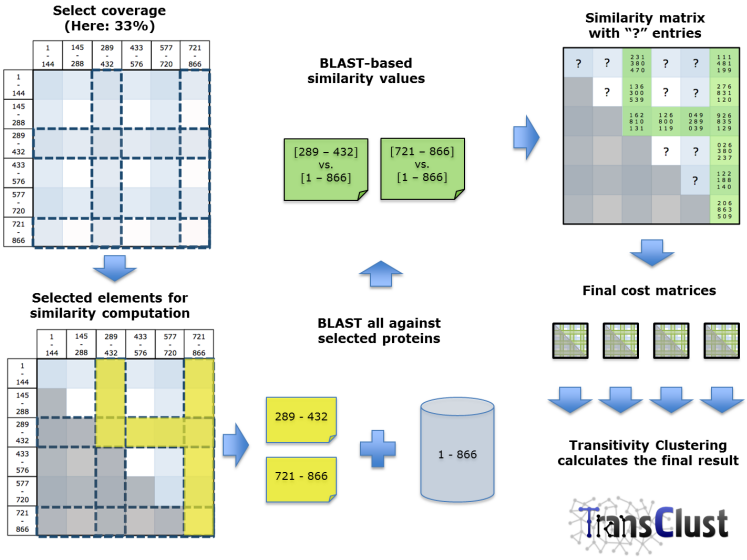


Figure 4.2.: Illustration of the block-based scheme for saving similarity value computation time. For illustrative reasons, we arbitrarily ordered the data to form six blocks. Now, only the data from block three and six are selected to be BLASTed against the database. The remaining values in the similarity matrix are displayed as "?". These are the steps used by the CMC to create cost matrices for TransClust.

of the Brown *et al.* dataset has a mean of $\mu = 164.5$ and a variance of $\sigma^2 = 9604$ whereas the inter-cluster distribution has a mean of $\mu = 0.352$ and a variance of $\sigma^2 = 6.995$. According to the model, choosing a threshold $t \in [2, 113]$ ensures that $P(10) < 0.05$. For $P(2) < 0.05$, the threshold should be picked from $[4, 50]$. In fact, the best clustering quality (in terms of the best F-measure when compared to the gold standard) with missing values is achieved when using a threshold between 22 and 24, depending on the number of missing values. Refer to Section 4.3 on the next page for details of the evaluation.

In summary, the model suggests that the existing similarities can account for the missing values. In the Section 4.3 on the following page, we will evaluate the TransClust with missing values with two datasets.

4.2.2. Cost-Matrix Creation

In the previous subsection, we demonstrated how to integrate explicitly missing values into the WTGPP. Once a threshold for the clustering using TransClust is chosen, all explicitly missing values are replaced by the actual threshold and all other parts of the clustering process remain unchanged. Hence the integration of explicitly missing values only requires changes in the creation of the cost-matrices. There is one restriction to this approach: during the cost-matrix creation, implicitly and explicitly missing edge weights must be distinguished. As it is already convention to leave implicitly missing similarities out, we now use "?" for indicating that an edge $uv \in E_E$.

Such an integration in the CMC is straightforward but requires the existence of such a similarity file.

We again integrate the support for the clustering of protein sequences based on a BLAST all-vs.-all run. In order to provide the user with the possibility to systematically benefit from missing values in a BLAST all-vs.-all run, we do not use missing values at randomly chosen positions in the similarity matrix, but omit the calculation of entire blocks of the similarity matrix. First, a BLAST database for all protein sequences V is created by the CMC. Afterwards, only a selected subset $V_E \subset V$ of the available proteins V are BLASTed against the database. Figure 4.2 depicts this process. In other words, instead of performing an entire all-vs.-all BLAST run, CMC performs "one-vs.-all" runs for V_E. V_E either can consist of randomly selected proteins or be manually created by the user in order to incorporate prior knowledge. This is a reasonable approach as BLAST queries protein sequences against a database structure. Thus, a "one-vs.-all" query has a significantly smaller runtime than the same number of subsequent "single pair" queries. The combination of TransClust with the novel CMC omitting the calculation of entire stripes of the similarity matrix is called TransClustMV (TransClust with randomly missing values) in the remainder of this thesis.

That approach also enables an efficient distinction between a missing edge $uv \in E_I$ and $uv \in E_E$. If an edge weight is missing in the similarity matrix, indirectly and explicitly missing values can be distinguished as follows:

$$uv \in \begin{cases} E_E & u \notin V_E \wedge v \notin V_E \\ E_I & \text{otherwise.} \end{cases}$$

Consequently, only the set V_E needs to be stored additionally to the similarity matrix instead of storing every explicitly missing edge $uv \in E_E$ in the similarity matrix with a "?". To summarize, this approach has four major advantages:

1. The user is able to utilize the well accepted standard NCBI BLAST.
2. The user saves the calculation time for the missing similarities.
3. As the missing values are not stored explicitly, the size of the similarity file remains small.
4. The created cost matrices can afterwards be integrated into existing TransClust analysis pipelines.

Also note that our approach would work with similarity functions other than BLAST as well. The user has to provide the similarity file and the set V_E used for creating the similarity matrix.

4.3. Results & Discussion

We utilize the gold standard dataset from Brown *et al.* [23] for studying the effect of missing values to the clustering performance, i.e., accuracy and runtime (for a descrip-

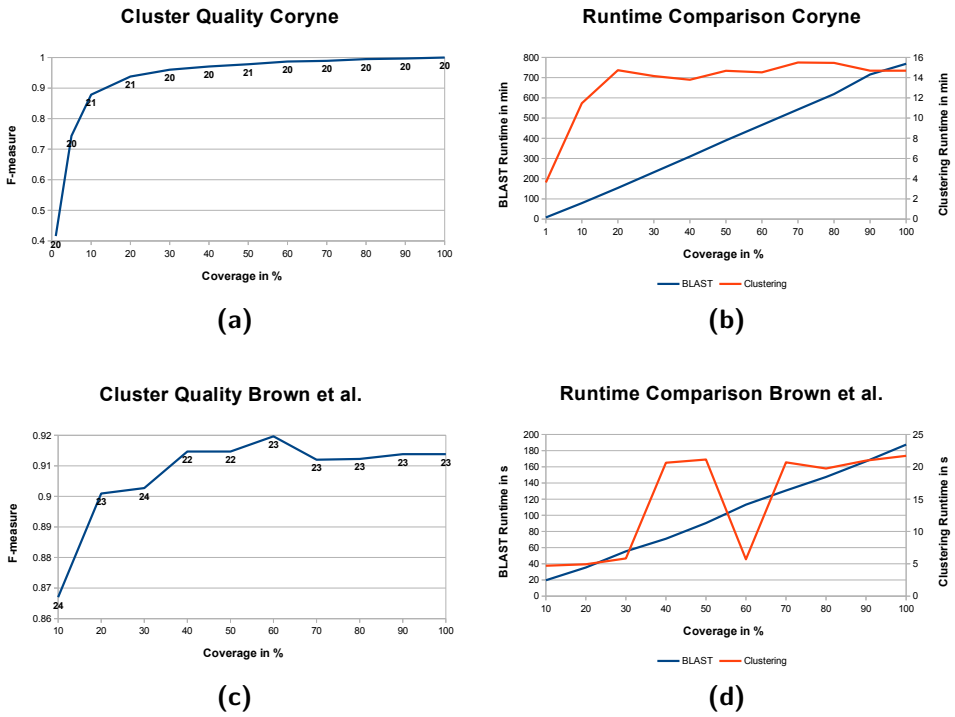


Figure 4.3.: The plots depict the runtime and accuracy of both datasets; the Coryne dataset is in the first row (**(a)** and **(b)**), and the dataset of Brown *et al.* is in the second row (**(c)** and **(d)**). Figures **(a)** and **(c)** display the F-measure as a function of the coverage. The numbers on the line indicate the threshold, which yielded to the corresponding F-meassure. Figures **(b)** and **(d)** give two runtime measures, again as a function of the coverage: blue displays the time consumed for calculating the BLAST results, and red depicts the runtime only for the clustering. Note that both runtime plots have different timescales displayed on the left for BLAST and on the right for clustering respectively.

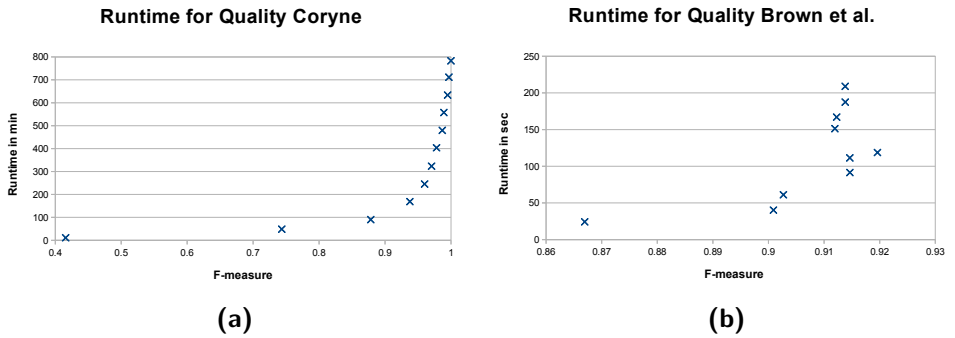


Figure 4.4.: Runtime of TransClustMV as a function of the F-Measure for **(a)** Coryne dataset and **(b)** Brown *et al.* dataset. Due to the small size of the Brown *et al.* dataset and the resulting short BLAST runs, we observe a certain variation of the clustering runtime. For the larger dataset, this is not observed anymore.

tion of the dataset refer to Subsection 2.6.1 on page 62). As the dataset from Brown *et al.* comprises a rather small problem instance, we cannot expect large improvements in terms of reduced computational time. Thus, we also apply our methods to a larger remote homology detection dataset consisting of 66,000 proteins of 27 different *corynebacteria*, a subsample of the Actino dataset described in Subsection 2.6.2 on page 62. In the remainder of this chapter, we refer to this dataset as the Coryne dataset.

We use the F-measure for comparing the results of TransClustMV to the above described gold standard. As the Coryne dataset has no gold standard, we compare the results of TransClustMV against the clustering result of TransClust with full information. For creating this "gold standard" we used a threshold of 20. For systematic evaluation, we vary the coverage from 1% to 90% (proteins used for one-vs.-all queries) and create the cost matrices accordingly. For each clustering, we compute the F-measure and measure the runtime. The runtime for the entire clustering includes both the time for creating the similarities and the successive clustering process. In order to assess the variability (robustness) of the best threshold, we clustered both datasets with different thresholds and always picked (and reported) the threshold leading to the best F-measure.

The results of the clusterings are depicted in Figure 4.3. We can see that even for a low coverage (high amount of missing values), the F-measure for both datasets only drops by a low percentage. Furthermore, it is important to notice that the threshold resulting in the best F-measure is very stable for all coverages. In conclusion, all methods for finding a good threshold (e.g., using a smaller gold standard dataset for parameter training or utilizing the same threshold from a comparable study) can be applied for clustering with missing values as well. The runtime scales as expected: BLAST computing times grow linearly with the number of sequence comparisons while the runtime for the clustering process is essentially constant with little variation. Figure 4.4 plots the average runtime of TransClustMV against the F-Measures. It shows that with our approach, the runtime can be drastically reduced, while the drop of the F-Measure is comparably moderate, i.e., less than a 10% F-Measure drop for about 70% runtime reduction. All runtimes are based on a single thread execution of BLAST and TransClustMV. Note that the novel CMC also supports parallel execution which further reduces the runtime.

Results of this Chapter

- Many similarities are redundant and do not effect the clustering result.
- TransClustMV enables the strategic exploitation of missing values.
- For protein homology detection, when we accept a quality drop of only 10% we can save $\approx 70\%$ runtime.
- Availability: `http://transclust.mpi-inf.mpg.de`

5. Active Transitivity Clustering

This section is based on the not yet published work:

Richard Röttger, Alexander Junge, Jan Baumbach. **Active Transitivity Clustering.** (working title, in preparation)

Objectives of this Chapter

- Development of strategies for judging the importance of missing similarities.
- Selective calculation of only those similarities promising the highest impact on the cluster quality.

In the last chapter we have demonstrated that omitting the calculation of randomly selected similarities reduces the overall calculation time while the cluster quality remains high even for small percentages of known similarities. The next step in exploiting missing values is to omit missing values not randomly but selectively, considering those values that are deemed most redundant. In other words, we aim to calculate only those similarities which carry the most valuable information for improving the cluster result. We aim to further reduce the amount of calculated similarities while improving the quality compared to the random approach. This enables the handling of large datasets and at the same time, the utilization of computationally expensive similarity functions.

In comparison to the approach presented in the previous chapter, active clustering requires the implementation of an entirely new clustering strategy rather than only a modified cost-matrix creation. The new overall strategy is depicted in Figure 5.1. First, the clustering process is started by calculating a small percentage of randomly chosen similarities which are used to calculate an initial clustering. This clustering is analyzed and the missing similarity values are ranked according to their estimated importance. The ranking of the missing information is the most crucial part of the entire process and will be discussed in detail in the following sections. In the next step, the highest ranking similarities are calculated on the fly and the current clustering

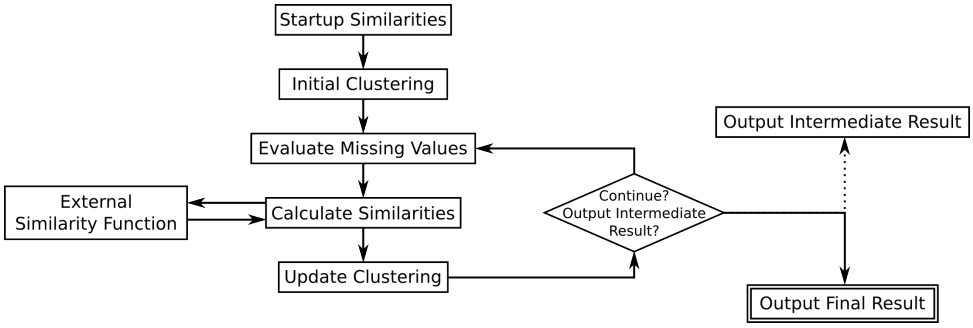


Figure 5.1.: The general active clustering approach: In the first step, initial similarities are calculated and an initial clustering is computed. The next step is the ranking of the missing similarities according to their likelihood to improve the cluster result. The most promising similarities are reordered in general by invoking an external program, e.g., BLAST. The current clustering is updated according to the new similarities. If no abort criterion is met (e.g., user interrupt, similarities are very unlikely to improve the result, etc.), the current clustering is reported to the user (if such an intermediate result is desired) and the process starts over with the evaluation of the missing values.

is updated using the new similarities. This similarity calculation and re-clustering cycle is repeated until certain abort criteria are met. As every clustering between the single steps is a valid clustering result, the user has the possibility to investigate these intermediate results and judge their quality. Thus, the user can already work on results based on these intermediate steps, while the clustering results are continuously improved.

The following sections discuss methods for judging the importance of missing values, introduce different active clustering strategies, and present the obtained results of the active clustering approach.

5.1. Evaluation of the Importance of Missing Values

5.1.1. Analyzing the Random Clustering Results

As seen in the previous section, the most important task is the development of a reliable method to judge the importance of missing values. In order to do so, we further investigate the results from the previous chapter. It is notable that the F-measure increases quickly initially and reaches a plateau; from there, only small improvements can be made while increasing the percentage of known similarities (refer to Figure 4.3 on page 83).

Figure 5.2 depicts the percentage of successfully reconstructed clusters depending on their size. We used the clusters of the Coryne dataset and utilized approximately 5% of all possible similarities[1]. We observe a clear bias towards the large clusters

[1]The Coryne dataset is compiled of 66,143 proteins resulting in 2.18 billion similarities. For the one-vs.-all queries, we used 1,654 proteins resulting in 108 million actually calculated similarities which correspond to a coverage of about 5%.

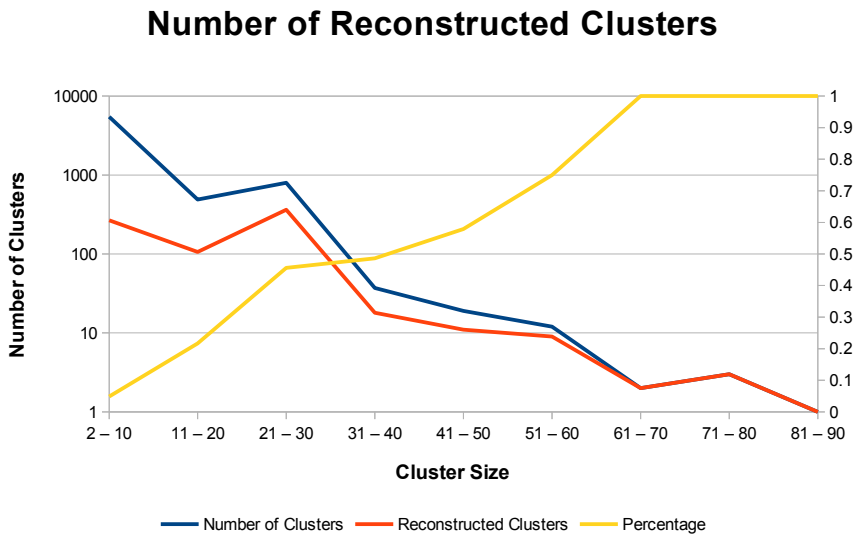


Figure 5.2.: The analysis of the clustering of the Coryne dataset with TransClustMV (5% of the similarities were calculated). The blue line indicates the number of clusters for a given cluster size of the result using full information (gold standard). The orange line shows the number of reconstructed clusters (overlap $\omega > 0.75$). Note the logarithmic scale for the "Number of Clusters" axis. The yellow line indicates the percentage of reconstructed clusters. One can clearly observe a higher reconstruction rate with growing cluster size.

which are almost always reconstructed correctly. In this context, "reconstructed" means we compare the reported clustering C with the gold standard clustering K. The gold standard is the result of a clustering with TransClust using full information. A cluster C_i is regarded as reconstructed, whenever there is a cluster K_j such that the overlap score $\omega(C_i, K_j) \geq 0.75$ (refer to Definition 2.3.6 on page 36). An overlap score $\omega(C_i, K_j) \geq 0.75$ prevents a cluster $K_j = \{u, v\}$ of size $|K_j|$ =2 from being counted as reconstructed when compared to a singleton $C_i = \{u\}$ (this would result in $\omega(C_i, K_j) = 0.5$). Also, clusters of size three are only counted as reconstructed when they are either completely matched or contained in a cluster of size at most four ($\omega(\{u, v, w, x\}, \{u, v, w\} = 0.75)$).

On the other hand, the performance for small clusters is comparably poor. This is apparent for clusters with exactly two elements. These two elements u and v are matched together if and only if the similarity $s(u, v)$ was calculated. With the random approach, this similarity is calculated with the probability of exactly the coverage, in this example, 5%.

To investigate that dependency further, we concentrate only on the detection of connected components for the remainder of this section. This is reasonable because TransClust only detects clusters within a connected component, thus the discovery of the connected components is the first crucial step towards the final clustering. To formalize the problem, let $G = (V, E)$ be a fully connected graph and $C =$

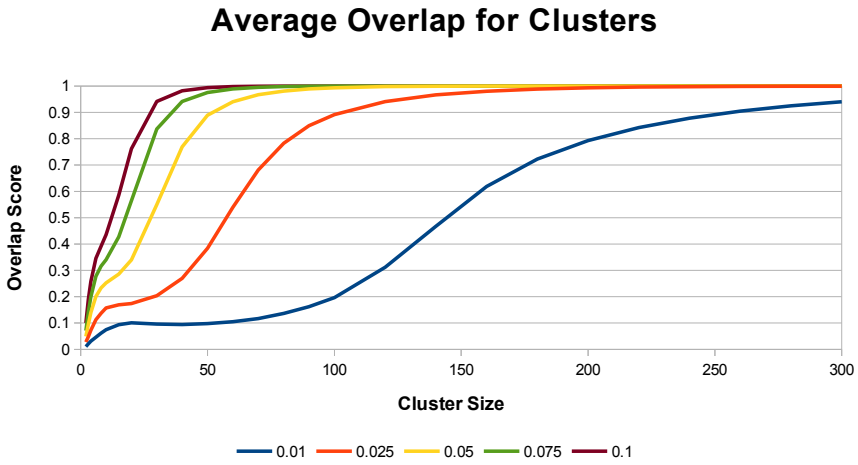


Figure 5.3.: Average overlap score between the artificially generated gold standard and the discovered connected components when using only a small sample of similarities (1%, 2.5%, 5%, 7.5%, and 10%).

$\{C_1, \ldots, C_n\}$ the clusters in the dataset with $s(u, v) \in [0, 1]$. We consider the ideal case in which all intra-cluster similarities are 1 and all inter-cluster similarities are 0, thus

$$s(u, v) = \begin{cases} 1 & \text{if } u, v \in C_i \\ 0 & \text{if } u \in C_i, v \in C_j \text{ with } i \neq j \end{cases}$$

holds true for all $u, v \in V$. We consider two objects belonging to the same connected component whenever their similarity is 1. If we now pick, with a probability of p, edges of the $\binom{|V|}{2}$ available edges, we ask how likely it is to discover a connected component of size c. This problem is a hard graph theoretical problem andto our knowledge, so far, has been solved only for graphs with the number of nodes growing asymptotically to infinity [43, 30, 88][2].

We want to provide an explanation for the problems with small connected components. As already mentioned above, for a connected component C_i with $|C_i| = 2$, the probability of detecting that component is exactly p. For connected components of size c with $c > 2$, the minimal required number of edges to be picked is $c - 1$, namely when picking a spanning tree. On the other hand, the total number of edges in that connected component is $\frac{c \cdot (c-1)}{2}$. That means that the minimal number of

[2]The study of random graphs was pioneered by Erdös and Rényi in the 1950s and has remained a subject of research since then. In their work, they defined a random graph $\Gamma_{|V|,|E|}$ as a graph with $|V|$ vertices and $|E|$ edges, which are uniformly chosen at random of the $\binom{\binom{|V|}{2}}{|E|}$ possible edges. They proved that the possibility of $\Gamma_{|V|,|E|'}$ being completely connected is given by $\lim_{n\to\infty} P(|V|, |E|') = e^{-e^{-2c}}$ with $|E|' = \frac{1}{2}|V| \log |V| + c \cdot |V|$ and c being an arbitrarily fixed number [43]. More recent work has been dedicated to allow for arbitrary node degree distributions, for example in [30, 88].

required edges grows only linearly whereas the number of "available" edges (and thus the number of picked edges with constant pick probability) grows quadratically. This intuitively explains that bigger connected components are more likely to be picked.

To empirically validate this observation, we created a dataset containing one cluster of each of the following sizes: $2, 3, 4, 6, 8, 10, 15, 20, 30, \ldots, 100, 120, \ldots, 300$. We ran the experiment picking 1%, 2.5%, 5%, 7.5%, and 10% of the edges with $10,000$ repetitions for each pick probability. The results are depicted in Figure 5.3. The data clearly shows the anticipated behavior that the larger the connected component, the higher the likelihood of being discovered. Furthermore, with increasing pick-probability, the size of the fully rediscovered clusters (average overlap greater than 0.75) shrinks to a cluster size of 20.

These results show where we can improve the results of the randomly missing values. We also need an effective method for tackling small clusters, even though the effects on the F-measure are relatively small. The remainder of the chapter first discusses how to judge the importance of missing values of big clusters, what we will now call "large critical clusters," and then suggest method to judge the importance of very small clusters, called "small critical clusters."

We will not give a sharp separation of a large and a small cluster. The reason for that reluctance is that in the remainder of this chapter, we will introduce different approaches in order to rank the importance of missing information. We designed methods which specifically are biased towards improving on small clusters and methods which tend to improve on large clusters. Nevertheless, these methods do improve the overall clustering, thus this "virtual" separation should only serve the purpose of easing the reading and not a strict separation of the set of clusters or the methods.

5.1.2. Large Critical Clusters

In this stage of clustering, we assume that we already have an intermediate clustering result, either based on the initial random similarity selection or due to previous iterations (see Figure 5.1). Let $C = \{C_1, \ldots, C_n\}$ denote the current clustering. As mentioned above, we want to extract those clusters of C which promise the most increase in the clustering quality when more similarities adjacent to these clusters are computed. Basically, we distinguish between two operations which can be performed with large critical clusters:

Split a large critical cluster into several smaller clusters

Join a critical pair of large clusters into one bigger cluster

In order to determine the most promising candidates for splitting, we must discuss the definition of the "criticality" of a cluster, i.e., the likelihood that cluster is split in the clustering with full information into several smaller clusters. The "criticality" of a cluster is defined as a relative measure resulting in a ranking of the available clusters. Let $\mathcal{Q}(C_i) : C \mapsto \mathbb{R}$ be a measure which assigns each cluster C_i a certain quality. As there is normally no gold standard at hand, we can only use internal measures.

Apparently, for calculating the measure, only the already known similarities can be taken into consideration.

Here, we suggest a measure based on the intra-cluster similarity. We consider those clusters as most critical whose average intra-cluster similarity is closest to the user-defined threshold t (refer to Subsection 2.4.10 on page 49 for an explanation of the threshold). Thus, $\mathcal{Q}(C_i)$ is defined as

$$\mathcal{Q}(C_i) = \frac{2}{|C_i| \cdot (|C_i| - 1)} \sum_{\{u,v\} \in \binom{C_i}{2}} s(u,v) - t$$

with $s(u,v)$ being the similarity function used for the clustering. We have already shown that the average intra-cluster similarity is always above the threshold [125], which also holds for clusterings with missing values as shown in [101]. Thus clusters close to the threshold, i.e., $\mathcal{Q}(C_i)$ close to zero, are exactly those clusters which are most likely to fall apart.

For the operation "Join," we need to consider pairs of clusters. The general approach remains similar; we now consider a measure $\mathcal{Q}(C_i, C_j) : \binom{C}{2} \mapsto \mathbb{R}$ which assigns every pair of clusters C_i and C_j a certain quality. In that case, it is the likelihood of these two clusters being joined together. Again, the measure $\mathcal{Q}(C_i, C_j)$ forms a relative measure, thus only comparing pairs of clusters with other pairs of clusters.

Analogous to the split operation, we base the measure on the inter-cluster similarity. The reasoning for the measure is that when the average similarity between two clusters C_i and C_j is only slightly below the threshold t, this pair is likely to be joined whenever new similarities are calculated. The measure is defined as

$$\mathcal{Q}(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{u \in C_i, v \in C_j} t - s(u,v).$$

Again, the closer $\mathcal{Q}(C_i, C_j)$ is to zero, the more critical the pair of clusters.

For both operations, the usage of any other internal measure is applicable as well, but using the similarities and the threshold of TransClust seems most promising as we can directly exploit the internal mechanisms of the WTGPP: Whenever the average similarity of one cluster drops below the threshold, the cluster necessarily falls apart to form an optimal solution of the WTGPP. The same is true whenever the average similarities between two clusters raises above the threshold; these two clusters in the current form cannot be longer part of the optimal WTGPP solution. We proved these properties of the WTGPP in [125]. With other measures, such a strong connection between the measure and the consequential change in the clustering cannot be established.

5.1.3. Small Critical Clusters

In the analysis of the results of the random missing values approach, we have seen that many small clusters remain undetected. In the active clustering approach, especially

after the first clustering steps, many objects are singletons, i.e., they do not yet belong to any cluster. If we consider the example of a two-element cluster, the similarity between these exact two elements must be known to join them into one cluster, which is not possible with the approach discussed in the last subsection as there are no inter-cluster similarities known between singletons.

The method we propose requires that entire stripes of the similarity matrix are calculated, the same principle as in the random missing values approach described in Section 4.2 on page 78. That is not a very harsh restriction, as many similarity functions (BLAST is among them) perform a one-vs.-all query more efficiently than single pairwise similarity queries. Let $S = \{s_1, \ldots, s_k\}$ be the objects for which all similarities are calculated and $U = \{u_1, \ldots, u_{N-k}\}$ with $N \gg k$ the remaining objects (compare to Figure 4.2 on page 81). In other words, every similarity $s(u_i, u_j)$ with $u_i, u_j \in U$ is unknown and all others are known. We now concentrate on the objects of small clusters of the set U. In order to receive a ranking of importance between the elements of U, we need to assess their expected similarity indirectly. Therefore, we construct a feature vector $\tilde{\mathbf{u}}_i \in \mathbb{R}^k$ for every object u_i:

$$\tilde{\mathbf{u}}_i = (s(s_1, u_i), \ldots, s(s_k, u_i)).$$

The vector $\tilde{\mathbf{u}}_i$ describes the "behavior" of the object u_i regarding the objects in S. The intuition is that two objects u_i and u_j of the same (not yet detected) small cluster show similar behavior, i.e., similar proximities towards the objects in S. Thus, all objects with a very similar vector $\tilde{\mathbf{u}}_i$ are candidates for belonging into the same cluster. The similarity $\hat{s}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{u}}_j) : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ between the vectors $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{u}}_j$ can be assessed, for example, with the Euclidean or the Manhattan distance. With the similarities $\hat{s}(\tilde{\mathbf{u}}_i, \tilde{\mathbf{u}}_j)$, a ranking of the most promising pairs of objects can be calculated.

An obvious drawback of this method is the necessity of calculating $\binom{|U|}{2}$ similarities between the objects of U and $|U| \cdot |S|$ features. That makes this method computationally challenging for large datasets. Furthermore, the actual similarity function $s(u, v) : V \times V \to \mathbb{R}$ must produce continuous values with a considerably high resolution, also in the case of dissimilar objects. When using a similarity function like the E-value of BLAST, most entries of $\tilde{\mathbf{u}}$ would be zero (because BLAST normally aborts the calculation in case of very dissimilar protein pairs), leading to unfeasible feature vectors.

5.1.4. Landmark Method

In 2011, Voevodski *et al.* published an approach Landmark-Clustering [116]. They also utilize a setting where "one-vs.-all queries" are actively exploited to generate an optimal K-median clustering. In their paper, they were able to show that they can reproduce a K-means clustering with only $\mathcal{O}(k)$ such one-vs.-all queries (with k being the number of clusters) if the problem instance resembles the (c, ϵ)-property of

Balcan *et al.*[3] [11]. This property basically ensures a "clusterability" of the problem instance using K-means. In other words, this approach only works on problems being favorable to K-means in the first place (i.e., well separated compact circle-shaped clusters).

The intuition of this method is to actively identify the potential centroids of the clusters (the authors call them landmarks) and utilize them for the one-versus-all queries. Their algorithm starts with a landmark selection process:

1. The first landmark l_1 is chosen randomly and added to the set of landmarks L.

2. The minimal distance from each point $v \in V$ to any landmark $l \in L$ is calculated: $d_{\min}(v) = \min_{l \in L} d(v, l)$.

3. Let $V' = V \backslash L$ be an ordered list of the remaining non-landmark objects such that for all $v_i, v_j \in V$ with $i < j$ holds $d_{\min}(v_i) < d_{\min}(v_j)$. The next landmark is randomly chosen from the q last (i.e., most distant) objects of V'.

4. Steps (2) and (3) are repeated until the desired number of landmarks is reached.

These landmarks are used in an adapted K-median algorithm to determine the final clustering.

This selection process ensures that the landmarks, i.e., the stripes in the similarity matrix, are well spread over the entire feature space of the given problem. For clustering with TransClust, the same principle can be applied, except that TransClust uses a similarity function as proximity measure. This approach also works best for large clusters, as small clusters can easily go missing because at least one member of a clusters has to be selected as landmark. The authors of [116] claim that besides the (c, ϵ)-property, the target clusters should be "large" but without specifying a concrete definition of large.

Furthermore, TransClust does not have a parameter k defining the number of desired clusters, thus there is no predetermined stopping criterion. Obviously, with an increasing sample over time, the selection will converge to the random missing value approach as at some point, there is no longer a meaningful most-distant object. As the sample strategy (calculating entire stripes of the similarity matrix) is the same as for the method for small critical clusters, both methods can be combined. Such an approach is presented in the following section.

[3] Definition of the (c, ϵ)-property as given in [11]: *"Given an objective function Φ (such as k-median, k-means, or min-sum), we say that instance $(\mathcal{M}, V)$ satisfies the (c, ϵ)-property for if all clusterings C with $\Phi(C) \leq c \cdot OPT$ are ϵ-close to the target clustering C_T for $(\mathcal{M}, V)$."* In this definition, $\mathcal{M}$ defines the metric space and V the objects in question. In other words, whenever a clustering is a c-approximation of the optimal clustering given the objective function Φ, it also limits the error compared to the target clustering (e.g., gold standard) to at most ϵ. The definition of error is not fixed, it can be for example the deviation of an external quality measure or the number of misclassified objects (see[11, 116] for more details).

5.2. Strategies for Active Clustering

5.2.1. Similarity Calculation Strategies

In the previous section, we developed methods to rank the importance of missing values. This section describes the actual implemented strategies in ActiveTransClust and their parameters. These strategies have been developed particularly to validate the different importance measures and will be discussed afterwards in the results section.

5.2.1.1. Intra/Inter-Cluster Strategy (IICS)

Intuition This method is based on the importance ranking of the "Large Critical Clusters" (Subsection 5.1.2 on page 89) and attempts to calculate those similarities which most likely lead to a merger of several clusters or to a split of a cluster.

Implementation Let E_E denote the set of edges for which no similarity was calculated so far, and E_K and E_I the edges for which similarities were calculated (compare to Section 4.2 on page 78). Given an intermediate clustering result $C = \{C_1, \ldots, C_n\}$ based on the similarities of the edges in E_K, the clusters and the pairs of clusters are ranked according to the method described in Subsection 5.1.2 on page 89. Let $R_C = (C_{i_1}, \ldots, C_{i_n})$ be the sorted sequence of all clusters of C according to their criticality, i.e., $\mathcal{Q}(C_{i_k}) \leq \mathcal{Q}(C_{i_l}) \quad \forall 0 < k < l \leq n$. Analogously, let $R_P = ((C_{i_1}, C_{j_1}), \ldots, (C_{i_m}, C_{j_m}))$ with $m = \binom{|C|}{2}$ be the sorted sequence of all cluster pairs according to their criticality, i.e., $\mathcal{Q}(C_{i_k}, C_{j_k}) < \mathcal{Q}(C_{i_l}, C_{j_l}) \quad \forall 0 < k < l \leq m$. From both lists, the top elements are selected for further investigation (parameter *numberOfClustersNewSimilarities*). In case of a critical cluster, the number of unknown similarities are calculated and a user-defined fixed fraction of them are calculated (parameter *fractionOfClusterSims*). The same strategy is applied to the critical pairs of clusters. Here, a fixed fraction (parameter *fractionOfClusterPairSims*) of missing inter-cluster similarities between the current cluster pair are selected. This straightforward implementation suffers two important shortcomings:

1. It can happen that after the initial clustering, there is no information at all between a pair of clusters (C_i, C_j), i.e., $uv \in E_E \quad \forall u \in C_i, v \in C_j$. Consequentially, this cluster pair cannot be ranked and thus would never be considered as a potential merge candidate. To overcome this problem, the user can specify a default criticality λ which these pairs are assigned (parameter *noInformationInterClusterQuality*), i.e., $\mathcal{Q}(C_i, C_j) = \lambda$. The closer λ is to zero, the more likely those cluster pairs are ranked on top in R_P.
2. Explicitly missing edges E_E and indirectly missing edges E_I need to be distinguished. In contrast to TransClustMV, they cannot be distinguished

by storing a list of objects V_E used for one-vs.-all queries but need to be stored completely (compare to Section 4.2 on page 78). That enlarges the similarity file in comparison to TransClustMV.

Parameters

numberClustersNewSimilarities The number of clusters/pairs of clusters for which new similarities are reordered. Higher ranking (pairs of) clusters are picked first.

fractionOfClusterPairSims The fraction of currently unknown intra-cluster similarities that is reordered for a pair of clusters that was selected.

fractionOfClusterSims The fraction of similarities that is reordered for each of the highest ranking clusters.

noInformationInterClusterQuality Average cluster-pair quality for a pair of clusters with no known inter-cluster similarities. A low value leads to preferred computation of similarities for these pairs of clusters.

5.2.1.2. Landmark Strategy (LS)

Intuition This method follows the landmark approach described in Subsection 5.1.4 using one-vs.-all queries. This method aims to distribute the one-vs.-all queries as efficiently as possible by choosing those objects as landmarks which are furthest away from the already known landmarks. This method is especially efficient if the used similarity function supports fast one-vs.-all queries (e.g., queries against a database or BLAST).

Implementation The algorithm starts with the one-vs.-all query of a randomly selected seed landmark. The object most distant to the current set of landmarks serves as the next landmark. Whenever several nodes are equally dissimilar to the set of existing landmarks, the new landmark is chosen randomly.

Parameters No additional parameters are required.

5.2.1.3. Extended Landmark Strategy (ELS)

Intuition The landmark strategy only takes the distribution of the landmarks over the entire space of objects into account but neglects the actual clustering structure so far. In other words, the landmark strategy might concentrate on the most distinct and easily detectable clusters as they are furthest away from all other data points but neglects small clusters, especially singletons. In Subsection 5.1.3, we introduced a strategy for ranking small clusters and singletons also based on one-vs.-all queries. The extended landmark strategy is the combination of both methods.

Implementation As already explained in Subsection 5.1.3, the ranking of the small clusters is computationally very expensive, as between all pairs of small clusters and singletons a similarity needs to be calculated. We implemented the small-clusters ranking (using the Manhattan distance) only as a post-processing step rather than a concurrent strategy to the landmark calculations. Let C^i denote the clustering after the similarity calculations for the ith landmark l_i. Given an internal quality measure $Q(C^i) \to \mathbb{R}$ or an external quality measure $Q(C^i, K) \to \mathbb{R}$ with K being the gold standard, we can define the notation of stagnation of the clustering quality: When the quality improvement of an iteration $Q(C^i) - Q(C^{i-1})$ is less than ϵ for k successive steps (paramter *numberOfSteadyIterations*), i.e., $Q(C^i) - Q(C^{i-(j+1)}) < \epsilon \quad j = 0, \ldots, k-1$, we call the clustering stagnating (same definition applies for the usage of an external quality function $Q(C^i, K) \to \mathbb{R}$). We utilize the F-measure when a gold standard is available, the silhouette value otherwise (see Section 2.3 on page 32 for details on the quality measures). Whenever the clustering quality is stagnating, we apply the extended step described in Subsection 5.1.3. In order to speed up the small cluster ranking, the algorithm takes only small clusters up to a user-defined size into account (parameter *maxClusterSize*). From the ranking of the missing similarities, *reorderVolume* values are calculated.

Parameters

ϵ, **numberOfSteadyIterations** Steady-state is reached when the F-measure or silhouette value does not change by at least ϵ over *numberOfSteadyIterations* iterations.

reorderVolume Number of similarities calculated for the "extended" step.

maxClusterSize Objects assigned to a cluster larger than this parameter are not considered in the ranking produced by the small-cluster ranking.

5.2.1.4. Summary

In the previous paragraphs, we introduced the actual implementation of the different strategies based on the ideas introduced in Section 5.1. Table 5.1 briefly summarizes the strategies. All these strategies have additional parameters controlling the actual execution of the program. The user can decide for a certain abort criterion (comparable to the parameters ϵ and *numberOfSteadyIterations* of the ELS) and whether intermediate clustering results should be reported and written as output or not. Furthermore, the user has the possibility to abort the clustering at any time and ActiveTransClust will report the current clustering as result.

The list of possible strategies is far from being exhaustive. There is the possibility of forming different combinations of these strategies and of course implementing different measures for judging the importance of missing values. The similarities queued for calculation are based on the ranking of only one measure. It might be beneficial to

	IICS	LS	ELS
Central Idea	Identify clusters/pair of clusters which are most likely to fall apart/be joined	Distribute one-vs.-all queries such that they most likely match cluster centroids	Same as LS, additionally account particularly for small clusters
Most important parameters	numberClustersNew-Similarities, fractionOfClusterPair-Sims, fractionOfClusterSims	no parameters	reorderVolume, number-OfSteadyIterations
Possible Drawback	Entire similarity file needs to be hold in main memory	No special treatment for small critical clusters	The "extended" step is computational ineffective

Table 5.1.: This table briefly summarizes the active clustering strategies implemented.

derive the ranking by means of a combination of different criteria (compare also to the Outlook Chapter 10 on page 159).

In the framework of this thesis, we understand the described methods as a proof of concept. The IICS allows the evaluation of the performance of concentrating on the "Large Critical Clusters" ranking. This method is also beneficial when there is an effective way of calculating single pairwise similarities. In contrast, if a one-vs.-all query yields performance advantages (in comparison to single queries resulting in the the number of similarities), the LS indicates the performance of exploiting this fact. The last strategy, ELS, will enable us to assess the influence of small clusters and the possible improvements by utilizing the "Small Critical Clusters" ranking.

5.2.2. Re-Clustering Strategy

After each iteration, the active clustering algorithm must create an updated version of the current clustering incorporating the new similarities. These additional steps pose a runtime disadvantage to normal clustering approaches and to TransClustMV. Consequently, active clustering can only be beneficial in terms of runtime saving when the time saved by omitting the calculation of the similarities outweighs the overhead for the clustering itself. In order to decrease that overhead, a re-clustering strategy must be developed which efficiently reuses the clustering obtained from the previous iteration.

All strategies allow to influence the number of new similarities calculated in each step (except for the LS). That can be used to influence the number of re-clusterings as the higher the number of calculated similarities per iteration is, the fewer iterations are required to reach a certain coverage of calculated similarities.

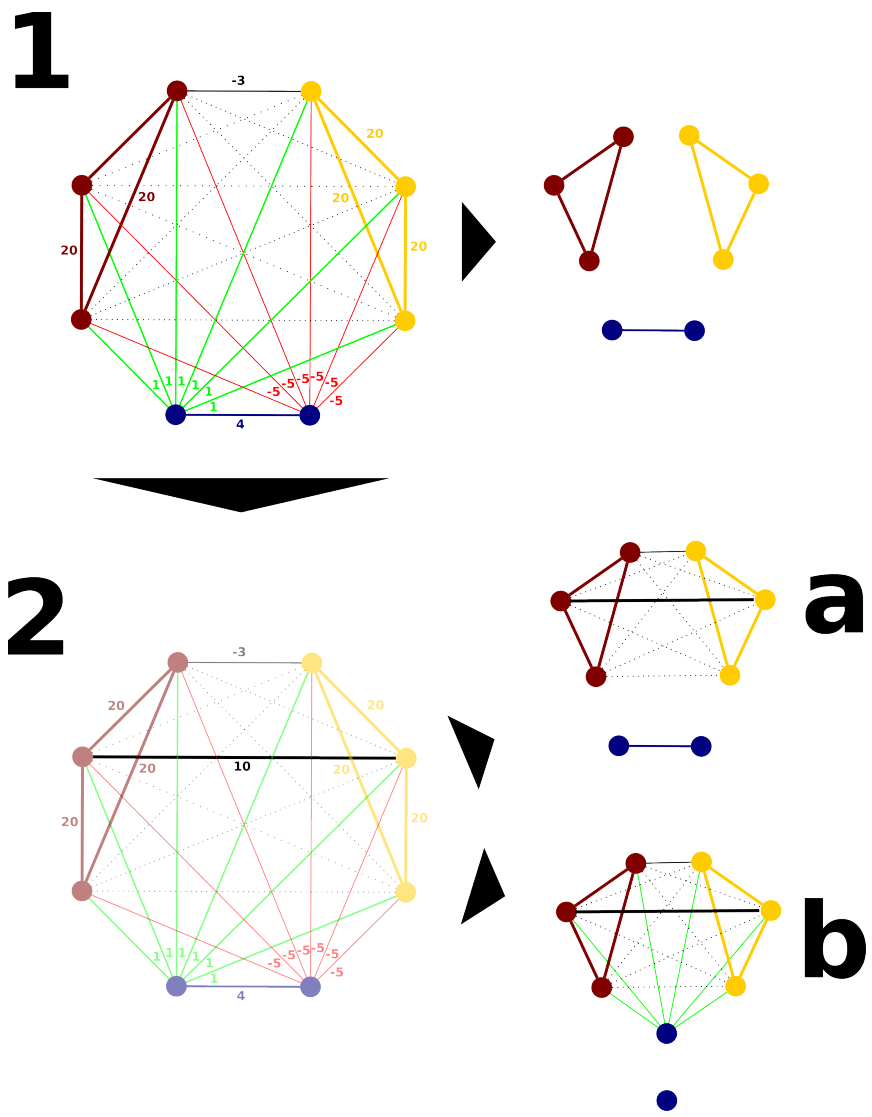


Figure 5.4.: Small example where the calculation of similarities between two clusters affects a third cluster. In this example, we use a threshold of 0 and unknown similarities are represented by dashed lines. In step **(1)**, the optimal solution for the given problem is displayed in the upper-right corner and consists of three clusters with a total solution cost of 6 (deletion of the green edges). In **(2)** an additional similarity is added between the brown and yellow cluster. When only the directly affected clusters of this new similarity are considered, the algorithm reports **(2a)** as the optimal solution (two clusters with total costs of 9). Nevertheless, the optimal solution is displayed in **(2b)** and requires to split the blue cluster (total costs of 7).

The next step is to reduce the computational effort for a re-clustering. As already seen in the previous chapters, TransClust basically performs two steps to calculate a clustering:

1. The dataset is split into connected components based on the known similarities
2. Each connected component is clustered

Splitting the dataset into connected components can be optimized in a straightforward manner. If a similarity value $s(u, v)$ with u and v belonging to the same connected component $\mathcal{C}_i$ is calculated, the connected component does not change. Only in case that $u \in \mathcal{C}_i$ and $u \in \mathcal{C}_j$ with $i \neq j$ and the similarity $s(u, v)$ is greater than the threshold t, the connected components $\mathcal{C}_i$ and $\mathcal{C}_j$ are merged. After updating the connected components, only those connected components require a re-clustering for which new similarities were reordered. On the other hand, these connected components can be very large and the total time spent in clustering is dominated by the largest connected component. Of course, the largest connected component is also most likely to receive updated similarities. Consequently, the above mentioned strategy has only limited influence on the overall clustering time.

When applying the IICS strategy, only similarities of high-ranked clusters (according to their criticality $\mathcal{Q}(C_i)$) or between high-ranked cluster pairs (according to their criticality $\mathcal{Q}(C_{i_k}, C_{j_k})$) are calculated in each iteration. In order to reduce the calculation effort for a re-clustering, we no longer re-cluster the entire affected connected components $\mathcal{C}_i$ but just the affected clusters C_j. Let $E^\star$ be the set of edges for which similarities were calculated in the current iteration. Let $C^\star = \{C_i \in C^k : \exists uv \in E^\star : u \in C_i \vee v \in C_i\}$ be the set of affected clusters of the current clustering C^k in the kth iteration. We construct "connected cluster components" CC which are maximal sets of clusters of which any two clusters are connected by a path of edges $uv \in E^\star$ with $s(u, v) > t$ (note, a connected cluster component can also consist of only a single cluster). Each of these connected cluster components is re-clustered in order to form the updated clustering C^{k+1}. In contrast to the previous optimization, this approach can no longer guarantee to result in the best possible clustering. Changing the internal structure of a cluster or a cluster pair by calculating new similarities can have side effects on other clusters (i.e., clusters with no new similarities) as well. Figure 5.4 depicts a small example using three clusters. In order to limit the divergence of the intermediate clustering from the actual best solution of the WGTPP over time, the user can specify a number of steps (*fastUpdateLength*) after which all of the affected connected components are re-clustered. This ensures that runtime is saved while still maintaining an optimal or near optimal intermediate cluster result.

To summarize, we are employing two different re-cluster strategies. The first method called "save re-clustering" ensures that TransClust delivers the best possible solution to the WTGPP:

1. Maintain an Union-Find data structure representing the connected components.

2. Let $E^\star$ be the edges for which new similarities are calculated, $\mathcal{C}$ the current connected components.

3. Update the connected components, i.e., union two connected components $\mathcal{C}_i$ and $\mathcal{C}_j$ iff $\exists uv \in E^\star : u \in \mathcal{C}_i \wedge v \in \mathcal{C}_j \wedge s(u,v) > t$.

4. Let $\mathcal{C}^\star = \{\mathcal{C}_i \in \mathcal{C} | \exists uv \in E^\star : u, v \in \mathcal{C}_i\}$ be the set of connected components affected by the new similarities. Cluster each $\mathcal{C}_i \in \mathcal{C}^\star$.

The second update strategy further reduces the re-clustering effort, but cannot guarantee that TransClust produces the best possible solution anymore. We call this strategy the "cluster-wise re-clustering":

1. Again, let $E^\star$ be the edges for which new similarities are calculated. Update the connected components as described above in steps 1 to 3.

2. Perform either a "full update" or a "fast update". After *fastUpdateLength* fast updates, a full update is performed:

   a) **Fast update:** Construct the "connected cluster components" CC as described in the text. Cluster each connected cluster component of CC.

   b) **Full update:** Cluster each affected component since the last full update.

In the evaluation of the different similarity calculation strategies in the next section, we only utilized the "save re-clustering" method in order to prevent any distortions from sub-optimal clusterings caused by the re-clustering strategy.

5.3. Results & Discussion

5.3.1. Biological Datasets

In order to assess the result quality of ActiveTransClust, we use the Brown *et al.* and Coryne datasets. For further information on these dataset, refer to Section 2.6 on page 62. As gold standard we use the result of a normal TransClust run with full information. Thus, in this chapter, a F-measure of 1 means that ActiveTransClust produced exactly the same result as the normal TransClust using full information.

As competition, we evaluate the different strategies against each other as well as TransClustMV. Figure 5.5 depicts the performance of the three different ActiveTransClust reordering strategies vs. TransClustMV. For the Brown *et al.* dataset, we can see that all three strategies perform better than the random approach of TransClustMV. Especially the performance leap of the ELS strategy is noteworthy. After calculating 6% of the similarities following the LS strategy, ELS performs the "extended" step which is responsible for the jump in the F-measure of over 10%. Except this peak, the quality performance advantage (in terms of a better F-measure) of ActiveTransClust over TransClustMV is about 5%. The Coryne dataset was only clustered with the LS strategy as the other strategies could not cope with the size

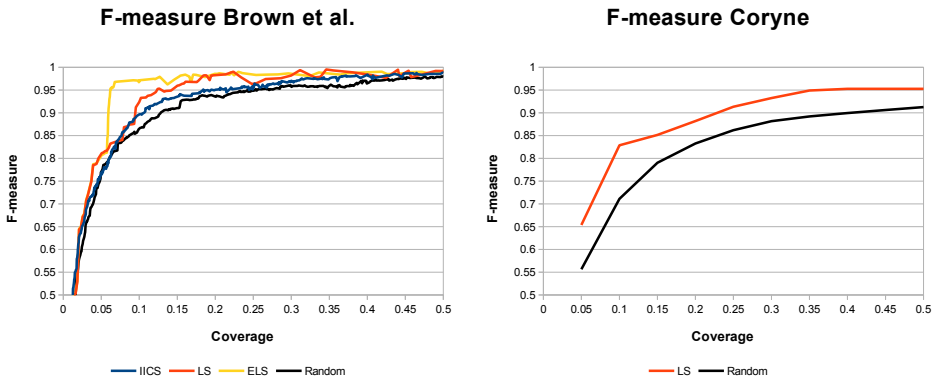


Figure 5.5.: Performance comparison of ActiveTransClust strategies IICS, LS, ELS, and TransClustMV (random) using the datasets of Brown *et al.* (left) and the Coryne dataset (right).

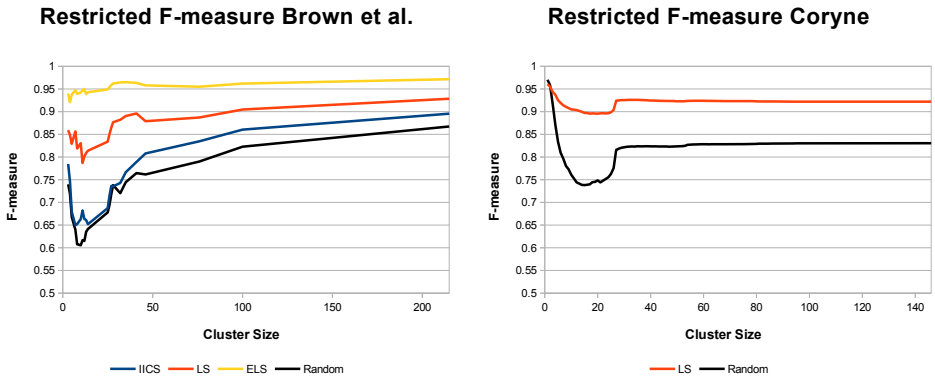


Figure 5.6.: Development of the F-measure with growing cluster-size using ActiveTransClust strategy IICS, LS, ELS, and TransClustMV (random). On the left side, the Brown *et al.* was used; the right side represents the results of the Coryne dataset. The F-measure is based on the clustering result utilizing 10% of all available similarities.

of the dataset. The ELS strategy requires the calculation and storage of all pairwise similarities for all objects of small clusters (compare to Subsection 5.1.3 on page 90). The IICS strategy necessarily needs the entire similarity matrix in the memory and cannot store explicitly and implicitly missing values as effectively as the other methods. This is the reason why ActiveTransClust was not able to produce results for that dataset using the IICS strategy. The remaining LS strategy performs better than TransClustMV on the Coryne dataset clearly indicating that the strategic selection of landmarks enhances the quality of the clustering result compared to the random approach.

Nevertheless, we can only observe a slight performance improvement over TransClustMV on both datasets. One reason is that few large clusters, which are efficiently reconstructed using any of the ActiveTransClust strategies or TransClustMV, dominate the F-measure. In fact, when only accounting for clusters up to a certain size (in the gold standard), we can see that the improvements over TransClustMV are more significant. Figure 5.6 depicts the F-measure as a function of the maximal cluster size. We calculated the clustering with a total of 10% of the similarities. We can clearly observe that TransClustMV performs much worse on small clusters than the active approaches. For the Brown *et al.* dataset, the IICS strategy performs similar to TransClustMV (which is expected, as IICS does not particularly account for small clusters), whereas the LS and ELS strategies show performance improvements. The ELS strategy shows almost a constant cluster quality over all cluster sizes which indicates that the treatment of the small clusters is important and works as intended. The F-measure only considering clusters $|C_i| < 10$ improved from 0.61 (TransClustMV) to 0.94 (ELS).

Even though the overall difference to the performance of the randomly missing values approach are rather small, the results are a strong indicator that the active clustering approach improves the cluster quality compared to TransClustMV using the same number of similarities. First, we have to consider that TransClustMV already performs well even when using only a small fraction of the similarities (compare to Subsection 2.2.5 on page 32). Second, the structure of the used datasets does not favor the active approach. For the Brown *et al.* dataset, only $\approx 19\%$ of the similarities are above the BLAST detection limit. The situation is even worse for the Coryne dataset where only $\approx 0.05\%$ of the similarities are above the BLAST detection limit. That means in turn that 99.95% of all similarities are indirectly missing edges E_i and thus set to the lowest possible similarity λ[4]. Even though we use more sophisticated methods to select the similarities compared to TransClustMV, chances are high that we select similarities of indirectly missing edges which cannot contribute significantly to unravel the actual structure of the dataset.

[4]The Brown *et al.* dataset contains 833 proteins (thus $\approx 345,000$ pairwise similarities) whereas the BLAST file stores only $70,000$ similarities. The Coryne dataset comprises of 66k proteins, resulting in a total of 2.2 billion similarities of which only 1.2 million are above the detection limit. That is also the reason why TransClustMV and ActiveTransClust can handle this dataset easily, as both methods do not need to store explicitly and implicitly missing values in the memory.

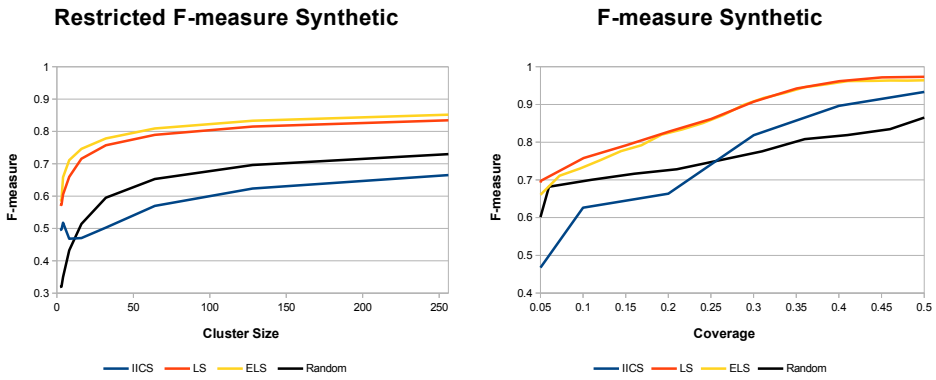


Figure 5.7.: Performance comparison of ActiveTransClust strategies IICS, LS, ELS and TransClustMV (random) using the artificially generated dataset. The left figure depicts the restricted F-measure at a sampling fraction of 20%.

The fact that the vast majority of all similarities are below the "detection limit" of BLAST also has consequences for the "Small Critical Clusters" ranking. Here on average, 99.95% of the elements of the feature vectors $\tilde{\mathbf{u}}_i$ are λ. Thus, selecting the next landmark, i.e., finding the most distant object to any of the existing landmarks, is based on the difference in the 0.05% similarities unequal to λ. We therefore use an artificial dataset with no such detection limit in the next subsection in order to see the potential of ActiveTransClust for non-BLAST similarity matrices.

5.3.2. Synthetic Datasets

Here, we follow the same evaluation process as with the biological datasets. The only difference is that we utilize a large artificially generated dataset. We generated this dataset following the dataset generator "Intra-vs.-Inter" described in Section 6.2 of the next chapter on page 115. In this case, we sampled the similarities from two normal distributions

$$s(u,v)\begin{cases}\sim\mathcal{N}(\mu,1) & \forall u,v\in V:\exists C_i\in C:u,v\in C_i\\ \sim\mathcal{N}(-\mu,1) & \text{otherwise}\end{cases}$$

with $\mathcal{N}(\mu,1)$ being the normal distribution. In this example, we used $\mu=2.5$ and a threshold of zero.

We created the dataset using the following cluster structure: we added 2^x clusters of size 2^{8-x} for $x=0..8$ to the dataset. That means we have clusters ranging from singletons to size 256. In order to reduce the bias of the F-measure, we have the same number of objects involved ($2^x\cdot 2^{8-x}=256$) in every cluster size, resulting in $9\cdot 2^8=2,304$ objects. TransClust with full information achieves an F-measure of 0.97.

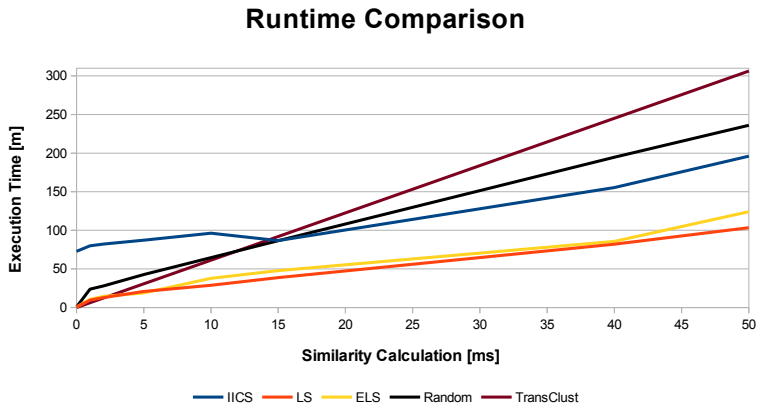


Figure 5.8.: Performance comparison of ActiveTransClust strategies IICS, LS, ELS and TransClustMV (random) using the Brown *et al.* dataset.

The main difference to the biological datasets is that this dataset possesses a real distinct value for each similarity and does not have most similarities set to a cut-off value λ. Figure 5.7 depicts the simulation results for the artificial dataset. Again, ActiveTransClust outperforms TransClustMV in the overall clustering result as well in the restricted F-measure. The worst performing active method is IICS which outperforms TransClustMV only at sampling fractions $> 25\%$, which was expected. The dataset was designed in such a way that the large clusters do not dominate this dataset (every cluster-size contributes the same number of objects to the dataset); thus, a method especially tailored for improving large clusters cannot benefit from the good performance on those large clusters to the same extent. The LS and ELS methods are consistently better than TransClustMV and deliver up to 20% better F-measures than the random sampling of TransClustMV. It is remarkable that the "extended" step of the ELS strategy does not seem to improve the clustering as profoundly as with the Brown *et al.* dataset. The ELS method can only improve on small clusters if they were falsely classified in the first place. As this dataset is quite well separated, the LS strategy and TransClustMV already perform quite well on these small clusters. Again, this can also be seen from the quite poor performance of the IICS method, which primarily improves on the large clusters. Thus, the "extended" step only confirms the current clustering of the small clusters but does not change them in a significant way.

5.3.3. Runtime Analysis

We have already seen that the active clustering approach can improve the clustering result significantly compared to the random approach using the same number of similarities. An obvious disadvantage of the active clustering approach is the necessity for calculating intermediate clustering results with which the potential importance of the missing values is estimated. That means in turn that the time saved by omitting

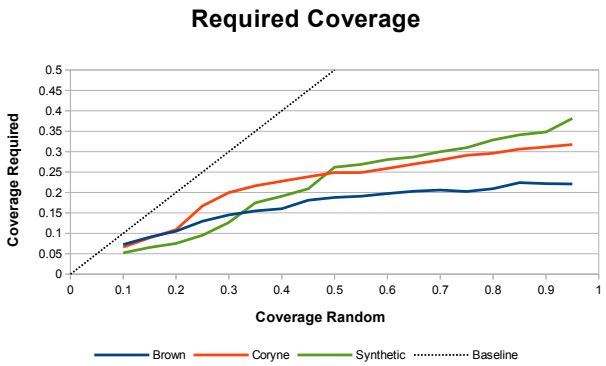


Figure 5.9.: This figure depicts the required coverage of the LS strategy (y-axis) to achieve the same F-measure as TransClustMV for a given coverage (x-axis). The dotted black line indicates the baseline, meaning no similarity saving.

the calculation of similarities should outweigh the additional calculation time for the more complex active approach in order to save runtime in comparison to TransClust.

Figure 5.8 depicts the calculation time of the different strategies of ActiveTransClust and TransClustMV in comparison to the normal TransClust. Here, we used the Brown *et al.* dataset and measured the required time of the different approaches until they reached an F-measure of 0.99 compared to the result of TransClust with full information. To show the effect of an increasingly more complex similarity function (in terms of computational time), we artificially delayed each similarity calculation a certain amount of time ranging from immediate reporting up to 50ms. The results show that even on a rather small dataset, the exploitation of missing values can be beneficial. The methods LS and ELS outperform TransClust already at a calculation of 4ms, the random approach and IICS require a delay of about $10 - 15$ms.

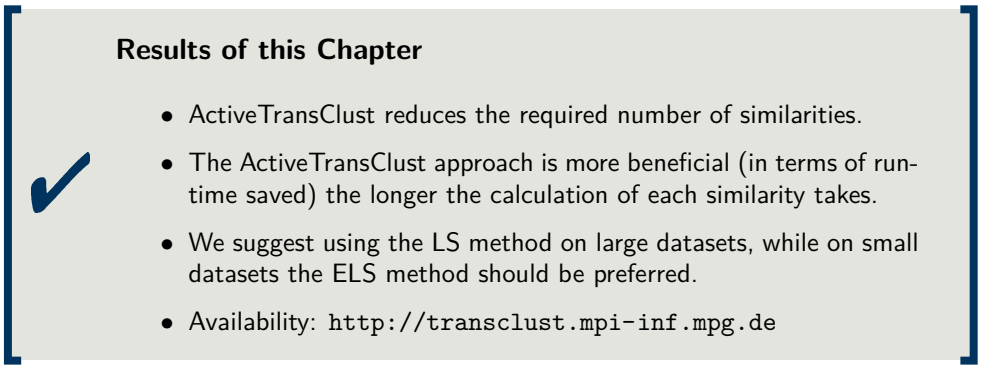
5.4. Conclusion

The evaluation of ActiveTransClust clearly shows an advantage over the clustering with randomly missing values. Especially on the synthetic dataset, the improvements were quite large. Figure 5.9 depicts the required similarity coverage to reach the same F-measure as TransClustMV. On all datasets, less than 30% of the similarities were required to outperform TransClustMV using 50% of the similarities. On the other hand, the evaluation of the biological datasets has shown that the application of ActiveTransClust might be subject to some restrictions. The similarity function should also be precise when two objects are dissimilar, and not only return a default value λ at a certain level of dissimilarity as e.g., BLAST does. With the artificial dataset where we did not have such a cut-off for the similarity function, we have seen that this information improves the effectiveness of the active clustering approach. Unfortunately, whenever using biological datasets, especially the "dissimilarities" are often only weakly documented. This is sometimes due to technical restriction but

often simply not of interest for the biologists creating these datasets (e.g. if an experiment shows that two proteins do not interact, such a "negative" result is very often simply not stored in the databases).

Furthermore, the benefits of ActiveTransClust are larger if the used similarity function is complex in terms of required computational time. The longer a calculation for a similarity takes, the more the user benefits from omitted similarity calculations in order to save running time in the overall clustering process.

To summarize, ActiveTransClust further reduces the required amount of similarity calculations while maintaining at least the same result quality in comparison to the randomly missing values approach.



6. ClustEval - A Cluster Evaluation Framework

This section is based on the not yet published work:

Christian Wiwie, Jan Baumbach, Richard Röttger. **Standardization and Evaluation of Popular Bioinformatics Clustering Tools - An Integrated Online Framework.** (in preparation)

Objectives of this Chapter

- The last chapters have demonstrated that clustering is complex and confusing for non-specialists.
- Unifying clustering approaches and data formats.
- Development of highly automatized clustering workflows guaranteeing bias-free high quality results for large-scale studies.

6.1. System Overview

The last chapters might have illustrated that performing a high-quality clustering study is a challenge as clustering is complex and several steps of an analysis have to work perfectly together in order to produce reliable results. Following the general design of a cluster analysis as described in Subsection 2.1.3 on page 25, the most important factors and tasks to be considered are:

- Nature and structure of the input dataset
- Choosing the best suited clustering algorithm
- Performing the appropriate preprocessing
- Choosing the correct similarity function
- Evaluate the results and optimize the algorithm's parameters

All these points above are challenging for the practitioner. As already mentioned, there is no clustering algorithm which is the best choice for all problem instances [83]. Many clustering methods exist, all using different approaches to the task, as well as different file formats for input and output (compare to Section 2.4 on page 39). In order to find a well suited clustering algorithm, a meaningful evaluation scheme needs to be applied. For the evaluation of the datasets and clustering results, a plethora of measures with different key aspects exist (compare to Section 2.3 on page 32). The choice of a quality measure is also highly problem specific [59]. Thus, it is often a practical choice to apply several clustering algorithms and quality measures to the same dataset [47] and modify each step of the cluster analysis depending on the result [59]. In order to apply several clustering algorithms to the same dataset, the researcher faces further problems: Most clustering tools have different standards for input and output, each algorithm's parameter need to be optimized for the problem at hand, etc.

In order to bring more structure to the numerous methods and approaches, we developed the clustering framework ClustEval. ClustEval eases these obstacles for performing a cluster analysis. The framework automates the simultaneous execution of several clustering tools and also performs an automated threshold probing for the clustering algorithms. Along with the framework we provide several representative datasets, gold standards, and clustering tools. These results are published on our website so the user can easily spot the similarities between his dataset and the provided datasets, and see the performance (and the corresponding parameter sets) of the different clustering tools on these datasets. This eases the decision of which tool, parameter set, and evaluation method to choose.

6.1.1. Highlevel Setup of the Framework

In general, the framework is separated into a back end, performing all calculations, and a front end, presenting the results as a website. In this thesis, we discuss the design of the back end. The structure of the back end is depicted in Figure 6.1. The back end is implemented in Java and utilizes Rserve[1] in order to facilitate the numerous R routines existing for different measures. The central part of the back-end is the server in combination with the repository. The repository itself is a file and directory structure on the file system and provides ClustEval with all available entities (for the actual structure refer to Appendix A.1 on page 181). These entities are oriented on the typical steps of a cluster analysis as outlined in Section 2.1.3. The following four entities are used in ClustEval:

1. Dataset
2. Clustering Algorithm
3. Statistic
4. Run

[1] http://rforge.net/Rserve/

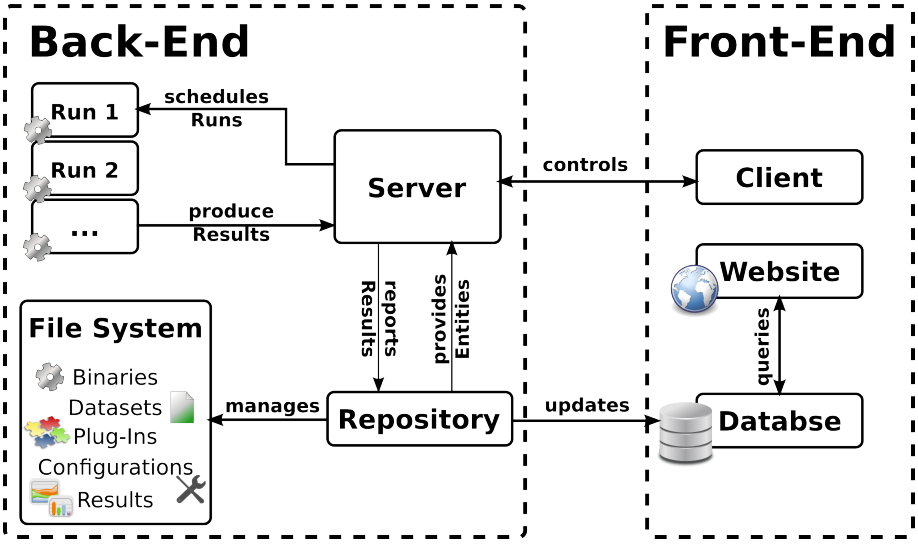


Figure 6.1.: Technical organization of the ClustEval Framework. The back end consists primarily of the server which manages all running processes of the back end. The Repository organizes all resources available to the server and also stores the generated results on the file system. As displayed, the front end consists of two parts: (1) the client which is used to control the server and (2) a presentation layer implemented as a website.

All entities are defined by means of a configuration file located in the appropriate folder of the repository. The first three entities can be regarded as static "provider" entities whereas the "run" entity represent the logic of the system. Figure 6.2 depicts the general interconnection between these entities. In the Appendix A.2 on page 183 examples for the different configuration files are given. The server automatically screens the repository for changes in the entities and loads newly added entities on demand.

Beside the management of the repository, the main task of the server is the invocation of the different clustering and analysis tasks. The server receives tasks and control instructions via the command line client. Once a clustering task is executed, the server organizes the interplay of the different entities and maintains an optimal load of the hardware. Once the processes have generated results, they are stored in the repository file structure for further analysis by the user. Furthermore, results stored in the repository are also stored in a MySQL database which is in turn queried by the web interface of ClustEval. This interface presents all results in a user-friendly manner to the researcher. The separation of the repository and the website by means of an extra database as mediator was implemented for greater flexibility. With that design, presentation and calculation are cleanly separated and can even run on different machines; the ClustEval server does not need to run in the background in order to host the ClustEval website.

In the remainder of this subsection, we describe different entities in more detail and explain the possible configurations.

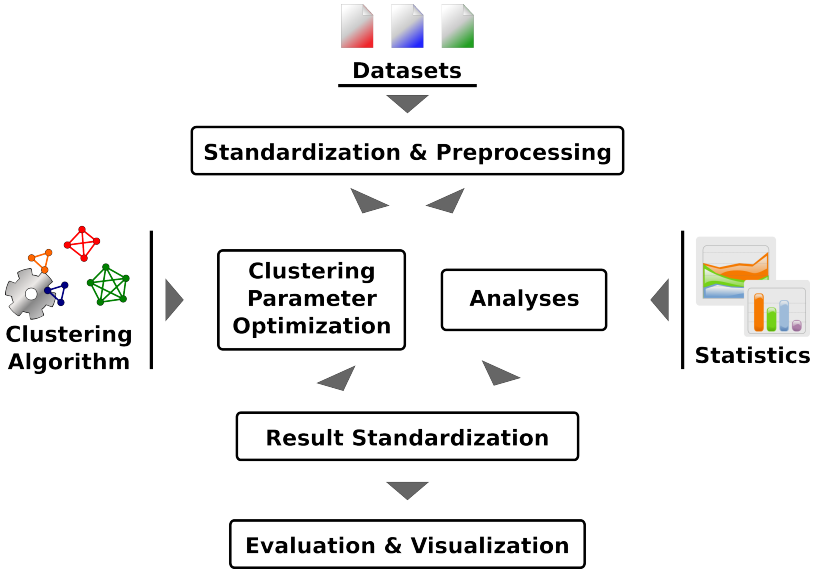


Figure 6.2.: Schematic overview of a cluster analysis performed by ClustEval. All tasks which are performed in an automated fashion are presented in a box; the boundaries of the framework are indicated by the solid lines. Cluster studies are eased by ClustEval by automatization and standardization. Data inputs are automatically converted to the standard input format, and calculated clusterings to the standard output format. For each clustering, many clustering quality measures can be evaluated automatically. When good density parameters are unknown for the clustering methods, parameter optimization can be performed in a highly automated fashion. Data inputs, as well as clustering results, can be analyzed and visualized on the website.

The Dataset Entity The *dataset entity* provides ClustEval with datasets. Every dataset consists of the actual data and a corresponding configuration file containing additional information. The most crucial information about a dataset is the file-format. Generally, we distinguish between *relative* and *absolute* datasets. In absolute datasets, objects are given as absolute coordinates of points located in a d-dimensional space; thus the information needs to be transformed into similarities for most clustering tools used in bioinformatics. The relative dataset already contains a proximity measure between pairs of objects.

The Clustering Algorithm Entity The *clustering algorithm entity* provides clustering algorithms to the framework. ClustEval is designed to work directly with the binary file of a clustering tool. This ensures that also closed-source algorithms and reference implementations, regardless of their programming language, can be utilized. Analogously to the configuration of the datasets, the clustering algorithms must be accompanied with a configuration file. This file contains the format of the system call for the binary and the required parameters. ClustEval supports four different types of parameters: integer, float, enumeration, and flags. For integer and float, the allowed range can be specified; for enumeration, the allowed item names (e.g., if a program allows the choice between

three different measures: `-m manhattan | euclidean | maximum`). Furthermore, for every clustering tool, the input and output format is specified, so the framework can convert the result into the ClustEval standard format to further process the results.

The Statistic Entity This entity represents a certain measure or statistic which is used in different occasions in the framework. Of course, this includes the cluster quality measures presented in Section 2.3 on page 32 in order to automatically assess the quality of the clustering or the agreement with a gold standard. Besides these commonly used measures, ClustEval also provides more specialized measures, for example co-occurrence matrices which count how often two objects were clustered together within a given parameter range or when using different clustering algorithms. Furthermore, several "dataset measures" are implemented. These measures solely analyze the dataset or gold standard without any prior clustering. Examples for these measures are the clustering coefficient or the node degree distribution. Appendix A.3 on page 185 lists all available measures.

The Run Entity This entity interconnects all other entities with each other. A run is represented by a configuration file and can basically be classified into two different types:

Analysis Run In an analysis run the user specifies a dataset and defines the measures which he wants evaluated on that dataset. This run does not include any clustering and is supposed to assist the user in exploring the nature of a new dataset.

Clustering Run In the clustering run, one or more datasets are specified together with one or more clustering tools. ClustEval applies each tool on each dataset with a given parameter set. If the user has not yet determined a feasible parameter set, he can also include automated threshold probing based on a method described in Section 6.3 on page 116. The threshold probing can be utilized in combination with a gold standard and an external measure or without a gold standard utilizing internal measures.

6.1.2. Data Formats

One main obstacle hindering users to easily apply several clustering algorithms on a certain dataset is that almost every clustering tool requires a unique input format and also reports the results in a unique output format. Here, we suggest a standard format for datasets, thus for input, as well as for clustering results. We provide converters which can convert a dataset given in the standard format into the input format of the most commonly used clustering approaches used in bioinformatics. The same is true for the clustering results. We designed the formats as simple yet as flexible as possible. Datasets in standard input and output format are stored as tab-separated plain text files.

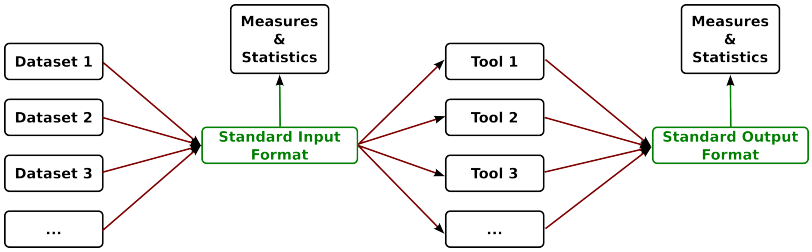


Figure 6.3.: This figure depicts the flow of a dataset through the ClustEval framework. Red arrows represent a data format conversion, green arrows indicate that the dataset can directly be used. Every dataset is converted into the standard input format. All measures are implemented to work on this format. In order to apply the given dataset to a clustering tool, the dataset is converted from the standard input format into the tool specific input format. The resulting output files is in turn converted back to a standard output format. That means, once a dataset is converted to the standard input format, all available tools and measures can be applied. When in turn for a new clustering tools converters for the input and output data are provided, this tool can be applied to all available datasets.

We suggest the following standards:

Input This format is a complete quadratic tab-separated similarity matrix with header row and column containing the ids of the objects.

```
1 id_1    id_2    ...    id_n
2 id_1    s(id_1,id_1)    s(id_1,id_2)    ...    s(id_1,id_n)
3 id_2    s(id_2,id_1)    s(id_2,id_2)    ...    s(id_2,id_n)
4 ...
```

In this file, `s(id_i,id_j)` refers to the similarity between the object `id_i` and `id_j`. This file format also allows for unsymmetrical similarity functions, thus for two objects $u, v \in V$ the similarity function s can produce results with $s(u, v) \neq s(v, u)$. In most cases, this design enlarges the file unnecessarily but ensures high flexibility.

Output The standard output format is designed to store the clustering result (with support for fuzzy clustering) together with the used parameter set. Let $\mathcal{P} = \{P_1, ..., P_l\}$ be the available parameters for the clustering tool in question, and $\mathfrak{P} = \{p_1, \ldots, p_l\}$ the actual values of the corresponding parameters. Furthermore, let $C = \{C_1, \ldots C_k\}$ denote the clustering result. Each cluster contains elements $C_i = \{c_{i,1}, \ldots, c_{i,s_i}\}$ with $s_i = |C_i|$. As ClustEval also supports fuzzy clustering (refer to Definition 2.1.9 on page 24), each object possesses a factor $f_{i,j}$ describing to which degree the element c_j belongs to cluster C_i. The format itself starts with a header naming the parameters, followed by the clustering result of every parameter set, each encoded in one line:

```
1 P_1,P_2,...,P_l
2 p_1,p_2,...,p_l          <C_1>;<C_2>;...;<C_k>
```

Here, <C_k> is replaced with a string representing each cluster. The cluster $C_i = \{c_{i,1}, \ldots, c_{i,s_i}\}$ is encoded as:

```
1 c_i_1:f_i_1,c_i_2:f_i_2,...,c_i_s_i:f_s_i
```

6.1.3. Extending the Framework

We developed the framework to be extendible by the users. The extension of the framework requires the implementation of a Java interface and copying the resulting binary file into the repository of ClustEval. The additional class-file is automatically detected and included by ClustEval. The most common extensions are briefly discussed below:

Dataset If the dataset is provided in a yet unknown format, the user must specify a new data format and implement a parser which transforms datasets of the new format into the standard input format. Afterwards, the dataset can be analyzed and clustered.

Statistic Adding a statistic (e.g., a quality measure) requires the user to implement the statistic only for the use of datasets in the standard format. All other data formats are converted to the standard format automatically.

Clustering Algorithm The user needs to provide a parser to convert the ClustEval standard input format to the input format required by the clustering tool. Furthermore, a converter is required to transform the result of the tool into the standard output format. Additionally, the user must provide a configuration file for the clustering tool such that the framework is aware of how to execute the binary with which parameters.

Even though the extension of the framework requires some additional work, the effort is drastically reduced in comparison to a manually processing of the tasks. If the dataset is already provided in the standard format or the clustering tool already accepts the standard format (or are provided in/accepts a format for which already parsers are available), no implementations are required for extending the framework.

6.2. Methods for Creating Artificial Datasets

A typical and important challenge in bioinformatics is the retrieval of datasets with a high quality gold standard. Gold standards are used in clustering for the determination of the best suited threshold or for the evaluation and comparison of the clustering performance (in terms of agreement with the gold standard, e.g., using the F-measure) of different tools. Generating gold standards for biological datasets is in most cases costly and time intensive as it normally requires wet-lab experiments coupled with a manual evaluation of the received data. For some problems, gold standards exist (e.g., the Brown *et al.* [23] for the classification of proteins into protein families).

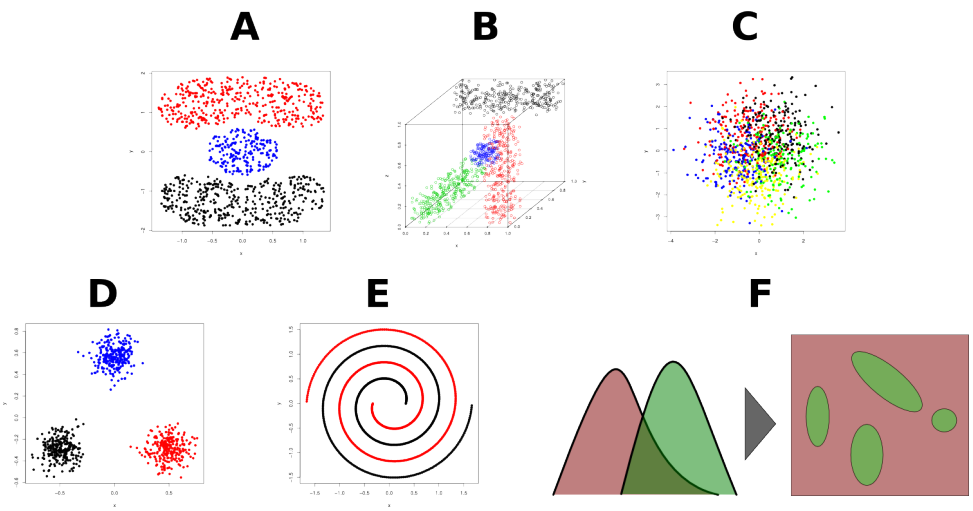


Figure 6.4.: Example datasets generated by the different artificial dataset generators. (A) Cassini dataset; (B) Cuboid dataset; (C) Gaussian 2D with 5 Gaussian curves and a very high overlap; (D) Simplex dataset in 2D; (E) Spiral dataset; (F) Intra-vs.-Inter dataset. The Intra-vs.-Inter dataset is only a schematic representation, as these datasets have no canonical projection into a two-dimensional space. On the left side, the two overlapping probability functions for the inter-cluster similarity (red) and intra cluster similarity (green) are displayed. The right side symbolizes the final dataset consisting of four clusters, where green indicates that the similarities were drawn from the green probability function and the space between clusters was sampled from the red function.

Nevertheless, even when a gold standard is available, further problems may arise. Gold standards of biological datasets are not necessarily complete. The fact that an element c_i is not a member of the cluster K_j in the gold standard $K = \{K_1, \ldots, K_m\}$ can be due to two different reasons: (1) the relationship of c_i to the cluster K_j was experimentally tested and a membership of c_i to K_j was inconceivable based on the evidence; or (2) the relationship of c_i to K_j was not yet exhaustively evaluated in the wet-lab, thus $c_i \in K_j$ can neither be guaranteed nor ruled out for certain. In the second case, if the clustering tool assigns c_i to K_j, it cannot be determined if that assignment is a false positive (for a definition of false positive in clustering refer to Section 2.3.2 on page 35). Treating those potentially false assignments as false positive may lead to an over-fitting of the clustering algorithm towards the true positives.

To summarize, even in the presence of a gold standard, it is difficult to accurately assess the quality of the cluster results and thus the performance of the clustering tools. "Entirely objective cluster validation is possible only on the data with known well-defined cluster structures and the development and evaluation of new clustering algorithms should therefore always include such data. In this context, the development of synthetic datasets that realistically mimic the properties of biological data are of particular importance as such an approach permits a controlled study of an algorithm's sensitivity with respect to specific data properties" [59]. Therefore, we integrated the following artificial dataset creators in ClustEval:

Cassini: The Cassini dataset is probably one of the most commonly used artificial datasets. It consists of three clusters, a circle in the middle and two Cassini-shaped[2] clusters bend around the circle. This non-convex shaped Cassini clusters pose a problem for algorithms favoring a certain shape of the clusters, e.g., for k-Means with the tendency to form circular clusters.

Cuboid: This is a three-dimensional dataset consisting of four clusters. Three cuboids are located at edges of the cubic space and one cube is placed in the middle of the space. This dataset is relatively easy to cluster but again penalizes clustering methods preferring spherical shaped clusters, as the clusters on the edges are elongated, box-shaped cuboids.

Gaussian 2D: In this two-dimensional dataset, the members of each cluster are distributed following a given two-dimensional Gaussian distribution. Here, the degree of overlap of the different clusters and the number of clusters can be varied by defining the according Gaussian distributions. Depending on the overlap and variance, this dataset can be very hard to cluster.

Simplex: This dataset generator places the clusters on the corners of a simplex[3]. Through the variance, the user can define the overlap between the different clusters but generally, the clusters in this dataset are very well separated.

Spirals: An often used two-dimensional dataset with two different clusters. The clusters are two entangled spirals. The correct clustering of this dataset is a very challenging task for most clustering algorithms. Due to the spiral form, both clusters spread over the entire plane and have a low cluster density (the average distance between two points of the cluster). Nevertheless, this dataset is more of academic interest, as it hardly resembles any known real-life dataset.

Intra-vs.-Inter: This dataset generator was specially intended to work with cluster algorithms using a similarity file as input. The number and size of the clusters can be arbitrarily chosen. The intra-cluster similarities are sampled from Gaussian normal distributions $X_+ \sim \mathcal{N}(\mu_+, \sigma_+^2)$, the inter-cluster similarities are sampled form $X_- \sim \mathcal{N}(\mu_-, \sigma_-^2)$ respectively. The user defines the mean μ_+, μ_- and the variance σ_+^2, σ_-^2 of both distributions.

Figure 6.4 depicts example datasets generated with ClustEval. The researcher can choose a data generator which fits a given real-world dataset best and test the different clustering tools on an artificial dataset with a given gold standard. The framework can also be easily extended with new data-generators.

[2]The Cassini oval is the set of points which quadratic distance between two given points remain constant. Given p and q, then for all points x_i of the Cassini curve, the following equation holds true: $d(x_i, p) \cdot d(x_i, q) = b^2$, where b is the parameter defining the actual shape of the curve.

[3]A simplex can be seen as the generalization of a triangle to an to arbitrary dimension. For example, a triangle is a two-dimensional simplex, a tetrahedron (pyramid with a triangular base) is a three-dimensional simplex. This simplex structure helps to distribute the different clusters evenly through the space in order to prevent biases.

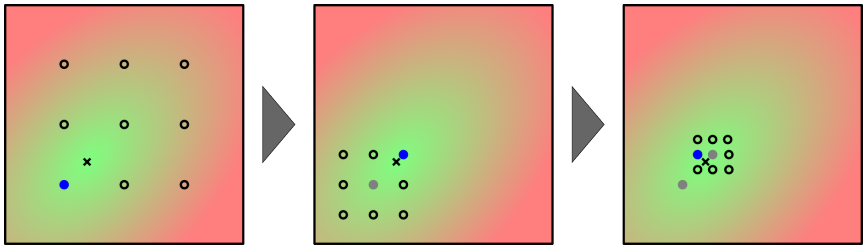


Figure 6.5.: This Figure illustrates the "Adaptive Divisive Parameter Optimization" method on a two-dimensional parameter-space. The background coloring represents the cluster quality as a heat-map (green for high quality, red for bad). In each step, the parameter set is selected (indicated as blue) and new parameters are probed on a smaller space centered around the previously selected parameter set. The small cross indicates the optimal parameter set.

6.3. Automated Threshold Probing

As already mentioned in the previous chapters, applying a certain existing clustering method to a given dataset is not straightforward. Besides the correct conversion into the tool's own data format, the tool itself must be used correctly with meaningful parameters. As seen in Section 2.4 on page 39, most clustering tools have parameters which are not directly connected to an interpretable property of the clustering result, but rather fine-tune some internal algorithmic details. To use these tools efficiently, expert knowledge about the algorithm is necessary.

The procedure of detecting a good parameter set can be separated into the following steps:

1. Select a quality measure $\mathcal{Q}(C) \to \mathbb{R}$ for assessing the clustering result quality.
2. Select a set of parameter sets $\mathcal{P} = \{P_1, \ldots, P_k\}$ which are evaluated.
3. Perform a clustering for each parameter set P_i.
4. Determine the best parameter set $P^\star = \text{argmax}_{P_i \in \mathcal{P}} \{\mathcal{Q}(C(P_i))\}$ with $C(P_i)$ being the clustering generated using the parameter set P_i.
5. *Optional:* Refine the set of parameter sets $\mathcal{P}$ based on the clustering results.

It is important to note that the parameters are optimized in order to maximize a given quality measure $\mathcal{Q}(C)$, i.e., the usage of different quality measures can lead to different optimal parameter sets.

In order to ease the usage of the different clustering tools and to enhance comparability of the different clustering approaches by means of a standardized threshold optimization, ClustEval provides several methods for automating steps 2-5. If a gold standard K is available, an external quality measure $\mathcal{Q}(C, K) \to \mathbb{R}$ can be used.

Grid-Based Parameter Optimization The range of each numerical parameter is covered by equidistant points. For Boolean values, both "true" and "false" are

tested. Each combination of these values forms the set of parameter sets $\mathcal{P}$. This method does not include any feedback mechanisms, and the number of combinations scales exponentially with the number of samples taken from each parameter.

Gap Statistic Parameter Optimization Clustering tools using the number of desired clusters as only parameter, e.g., k-Means, can benefit from previously performed work on the estimation of the optimal number of clusters in a dataset. ClustEval supports the *gap statistic*, introduced by Robert Tibshirani *et al.* in 2001 [111] The gap statistic compares the intra-cluster distance of the clustering results for varying k (i.e., the number of reported clusters) with the expected intra-cluster distances of a clustering of a reference dataset. Such a reference dataset is generated by sampling each reference feature uniformly over the range of the observed features in the actual dataset. To be formally correct, let us assume a dataset V with $|V| = N$ and $C = \{C_1, \ldots, C_k\}$ being a clustering with k clusters. First, the authors in [111] define a measure of compactness of the entire clustering by

$$\begin{aligned} D_r &= \sum_{u,v \in C_r} d(u,v) \\ W_k &= \sum_{r=1}^{k} \frac{1}{2|C_r|} \cdot D_r \end{aligned}$$

where $d(u,v)$ denotes a distance measure. When using a distance measure, the compactness is a monotonic decreasing function with growing k, as each newly introduced cluster decreases the variance of the clusters. The gap statistic is now calculated as

$$\mathrm{Gap}_N(k) = \mathbb{E}_N^*[\log(W_k)] - \log(W_k)$$

where $\mathbb{E}_N^*$ denotes the expectation of $\log(W_k)$ using N samples of the reference dataset (generated as described above). The expected compactness of the clusters based on the reference dataset is supposed to decrease slower than the compactness of the clusters of the real dataset. The optimal k_{opt} is where the gap between expectation and observation reaches its maximum:

$$k_{\mathrm{opt}} = \arg\max_k \{\mathrm{Gap}_n(k)\}.$$

Adaptive-Grid-Based Parameter Optimization This method represents an iterative approach to detect the best threshold for a given quality measure. The idea is to concentrate the parameter set search to the most promising area of the entire parameter space. It starts with a very coarse grid for the entire parameter space, using only three values for each numerical parameter. Let $\mathfrak{P} = [P_{\min}, P_{\max}]$

Name	Alias	Parameter	Range	Version
Affinity Propagation	AP	preference	$[min(S), max(S)]$	16/02/07
Hierarchical Clustering	HC	#clusters	$[1, n]$	R, 2.15.1
K-Means	KM	#clusters	$[1, n-1]$	R, 2.15.1
Markov Clustering	MC	inflation	$[1.0, 10.0]$	12-068
Transitivity Clustering	TC	threshold	$[min(S), max(S)]$	1.0

Table 6.1.: The clustering methods integrated into ClustEval. Every clustering method has at least one density parameter. Here we only show the main parameter. For each of those parameters, we indicate the valid value range. $min(S)$ and $max(S)$ are minimal and maximal similarities of the input similarity matrix S, n denotes the number of objects in the given dataset. The k parameter denotes the number of clusters.

be the parameter range of parameter P. The three parameter values p_1, p_2, p_3 are chosen as

$$p_i = P_{\min} + i \cdot \frac{P_{\max} - P_{\min}}{4}.$$

After each iteration, all parameter ranges are halved and centered around the best performing value. Formally, letting p_i be the best value of the best performing parameter set, the new parameter range $\mathfrak{P}_{\text{new}}$ is defined as

$$\mathfrak{P}_{\text{new}} = \left[p_i - \frac{P_{\max} - P_{\min}}{4}, p_i + \frac{P_{\max} - P_{\min}}{4} \right].$$

This process is repeated until a predefined number of iterations or parameter sets have been tested. Figure 6.5 depicts the scheme on a small two-dimensional example. This method drastically reduces the number of required parameter sets compared to the "Grid-Based Parameter Optimization Method" as it concentrates the parameter set samples on the part of the parameter space yielding the best clustering qualities. Nevertheless, the adaptive method has some restrictions compared to the non-adaptive method and cannot be applied to all clustering methods equally well. It is apparent that the adaptive approach can potentially be stuck in local maxima. In order to prevent suboptimal solutions, the behavior of the quality in dependency of the parameters should be tested, for example on a small sub-sample of the original dataset using the simple "Divisive Parameter Optimization" method. Figure 7.8 on page 148 depicts the behavior of the parameter of TransClust as an example for a well-suited application for this method.

	Case Study I	**Case Study II**	**Case Study III**
Clustering Methods	AP, HC, KM, MC, TC	AP, HC, KM, MC, TC	AP, HC, MC, TC
Datasets	Cassini, Cuboid, Spiral	Bone Marrow	Brown
Distance Measure	Euclidean	Euclidean, Pearson, Spearman	BLAST
Clustering Quality Measures	DI, DBI, SV, FM, JI, RI, Sn	DB, DBI, SV, FM, JI, RI, Sn	DB, DBI, SV, FM, JI, RI, Sn
Parameter Optimization Method	Adaptive Divisive	Adaptive Divisive	Adaptive Divisive
Optimization Iterations	1001	1001	1001
Optimization Parameters	see Table 6.1	see Table 6.1	see Table 6.1
Optimization Criterion	F-measure	F-measure	F-measure

Table 6.2.: Overview of the run configurations for the three case studies. In case studies I and III, we created and performed one parameter optimization run with the denoted settings. The bone marrow dataset clustered in case study II consisted of absolute coordinates; we utilized three different distance measures to convert those to pairwise similarities. Thus we have three datasets and for each, we created one parameter optimization run. The abbreviations for the quality measures are: DB - Dunn Index; DBI - Davies-Bouldin Index; SV - Silhouette Value; FM - F-measure; JI - Jaccard Index; RI - Rand Index; Sn - Sensitivity (see Section 2.3 on page 32)

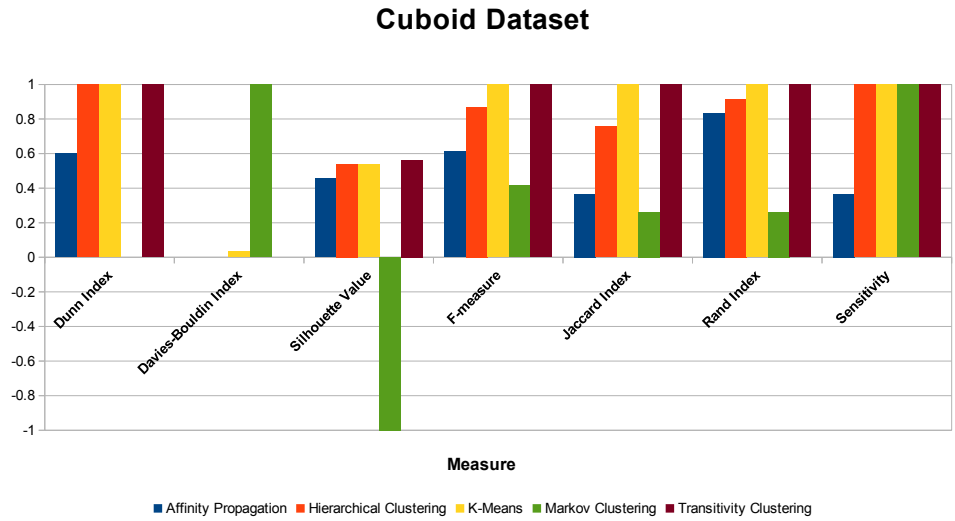


Figure 6.6.: This Figure shows the archived cluster quality measures for the Cuboid dataset. The values for the Dunn Index are scaled to $[0..1]$ as the Dunn index range is not bound.
This figure was taken from the ClustEval website and slightly modified to fit the style of the thesis.

6.4. Results & Discussion

In this section we want to demonstrate the power of ClustEval by means of three selected case studies. All case studies are based on a clustering task and are carried out using several clustering approaches, i.e., Affinity Propagation, Hierarchical Clustering, K-Means, Markov Clustering, and TransClust. Refer to Section 2.4 on page 39 for a detailed description of the different tools. Furthermore, for each clustering tool, a parameter optimization is performed. For every clustering method we specified which of its density parameters to optimize, and which parameter values to evaluate for the parameters respectively (refer to Table 6.1). Table 6.2 gives an overview of the run configurations used for the different case studies. We ran Markov Clustering with normalized similarities between 0.0 and 1.0.

In the following, we describe the case studies in detail and discuss their results.

6.4.1. Case Study I: Cluster Evaluation with Synthetic Data

In the first case study, we want to assess the cluster quality of the different clustering tools using several dataset. As already mentioned in the introduction, real-world datasets can pose the problem of an over-fitting as most of the time false negatives cannot be distinguished from true negatives. To overcome this problem, we utilized the dataset-generating capabilities of ClustEval. In this case study, we generated a Cassini, Cuboid, and Spiral dataset (refer to picture 6.4 A,B, and E). Here, Cassini and Cuboid resemble more natural problems, whereas the Spiral dataset was mainly

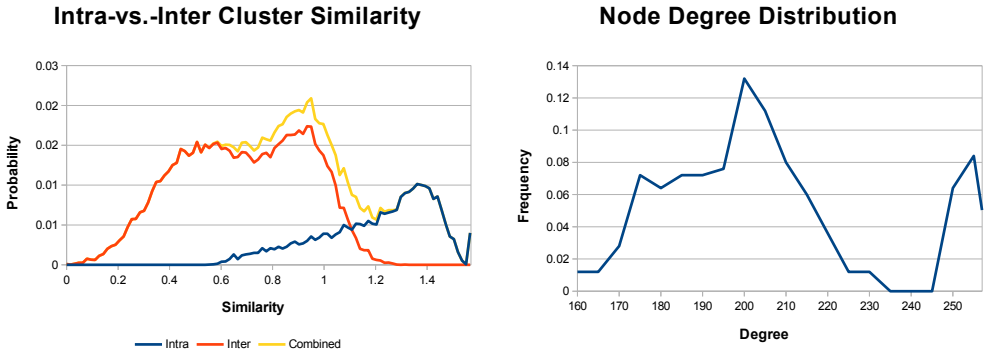


Figure 6.7.: This figure displays the statistics calculated for the Cuboid dataset. On the left side, the intra and inter cluster similarity is displayed. The overlap between both curves is relatively small which is an indicator of the general separability of the dataset. On the right, the node-degree distribution is displayed. In that case, it is the combined weight of all adjacent edges of an object.
This figure was taken from the ClustEval website and slightly modified to fit the style of the thesis.

Property	Value
Intra-vs.Inter Cluster Similarity	Figure 6.7
Overlap Intra-vs.-Inter	0.094
Similarity Distribution	Figure 6.7
Graph Adhesion	248
Graph Cohesion	248
Graph Min-Cut	101.38
Clustering Coefficient	1.00
Node Degree Distribution	Figure 6.7
Graph Density	1.00
Graph Diversity	0.99
Matrix Rank	250

Table 6.3.: Summery of the calculated properties of the Cuboid dataset.

chosen because the two entangled spirals are difficult to cluster for most clustering algorithms.

We created a clustering run with parameter optimization using the settings shown in Table 6.2. Additionally, we created an analysis run, assessing the following properties for each dataset: (1) The Intra-vs.-Inter similarity distribution, (2) the overlap of intra-inter similarity distribution, (3) similarity distribution, (4) graph adhesion, (5) graph cohesion, (6) graph min-cut, (7) clustering coefficient, (8) node degree distribution, (9) graph density, (10) graph diversity, and (11) matrix rank. Refer to the appendix A.3 on page 185 for an explanation of the measures.

In the remainder of this subsection, we concentrate on the results of the Cuboid dataset, as they showed the most interesting results. The results of the other datasets can be found in the Appendix A.4 on page 186. The Cubic dataset was created with a total of 250 points. Since the clusters are convex and isolated, we assumed this dataset to be easy to cluster. Furthermore, other measures for the dataset suggest that this dataset is very feasible for a cluster analysis, e.g., the intra-vs.-inter cluster similarity distribution or the clustering coefficient of 1.0. Figure 6.7 and Table 6.3 depict all calculated properties of the Cuboid dataset. For the other datasets, refer to Appendix A.4 on page 186.

Surprisingly, several clustering methods were not capable of separating the 250 data points into the four clusters. Figure 6.6 on page 120 visualizes all achieved clustering qualities on the cuboid dataset. TransClust and K-means were the only clustering methods solving the problem exactly according to the gold standard with an optimal achieved F-measure of 1.0 (TransClust was archiving the best F-measure on all three artificial datasets among the tested tools). They are followed by Hierarchical Clustering (F-measure of 0.869), Affinity Propagation (F-measure of 0.613), and Markov Clustering (F-measure of 0.415). Affinity Propagation turns out to be quite insensitive to different choices of its preference density parameter; the best achieved qualities are much worse than those of the competitive methods.

The silhouette value is around $S \approx 0.5$ for all methods except Markov Clustering. Its silhouette value of -1.0 is due to the fact that it generated only clusterings with one cluster for which the silhouette value is undefined and ClustEval returns the minimal quality. Since the silhouette value is defined in $[-1, 1]$, values of 0.5 are reasonably good. This points to the fact that the silhouette value is well suited for this type of clusters, since its assumption is that points of different groups are farther apart than points of the same group. The Davies-Bouldin index is close to 0 for all tools, except Markov Clustering. Analogous to the silhouette value, the Davies-Bouldin index cannot be calculated for only one cluster. We represented this fact in Figure 6.6 on page 120 by assigning 1 to the index for Markov Clustering (a Davies-Bouldin index of 0 is ideal).

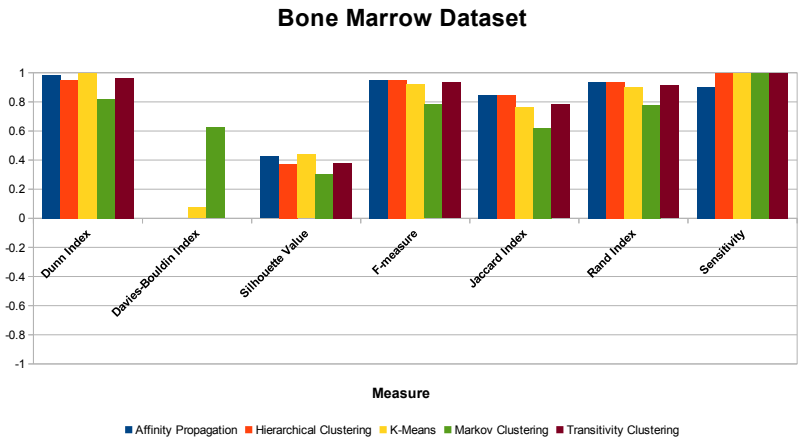


Figure 6.8.: This Figure shows the archived cluster quality measures for the Bone Marrow dataset. The values for the Dunn Index are scaled to $[0, 1]$ as the Dunn index does not have a bound values range.
This figure was taken from the ClustEval website and slightly modified to fit the style of the thesis.

6.4.2. Case Study II: Detecting Leukemia Subtypes in Gene Expression Levels

We used our framework to cluster a gene expression dataset containing 38 bone marrow cell samples of three acute leukemia sub-types [54]: 20 Acute lymphoblastic leukemia in B-Cells (ALL-B), 8 Acute lymphoblastic leukemia in T-Cells (ALL-T), and 10 Acute myelogenous leukemia (AML). We used ClustEval to assess the performance of several clustering methods and to identify the clustering method and respective density parameter values which clustered the dataset optimally according to the gold standard at hand (division into ALL-B, ALL-T, and AML).

We used ClustEval to calculate pairwise distances of samples in the dataset using three different distance measures and clustered each resulting similarity dataset independently. For each resulting dataset, we created one parameter optimization run, with the settings shown in 6.2. Best clusterings results were achieved on the Spearman Correlation dataset followed by Pearson Correlation and Euclidean distance.

In the following, we present clustering results using the Spearman correlation as they lead to the best clustering results. Figure 6.8 depicts all qualities using the Spearman correlation. No clustering method achieved a perfect separation of the dataset into the three leukemia sub-types, but overall achieved qualities (with the exception of Markov Clustering, all F-measures are≥ 0.90) were indicating that the cancer sub-types are well reflected by the gene expression levels, and the chosen distance measure preserves the information content. The best clustering in terms of maximal F-measure was achieved by Affinity Propagation and Hierarchical Clustering (0.948), followed by TransClust (0.933), KMeans (0.921), and Markov Clustering (0.783). The corresponding clusterings of Affinity Propagation, Hierarchical Cluster-

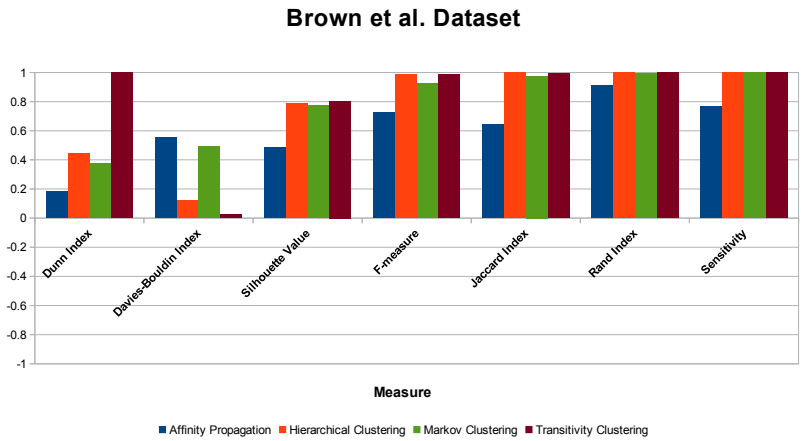


Figure 6.9.: This Figure shows the archived cluster quality measures for the Brown *et al.* dataset. The values for the Dunn Index are scaled to 1 as the Dunn index does not have a bound values range.
This figure was taken from the ClustEval website and slightly modified to fit the style of the thesis.

ing, K-Means, and Spectral Clustering contained three clusters which reflected the three sub-types with only a few misclassifications. Markov Clustering produced two clusters where the two ALL sub-types were merged into one cluster. TransClust also clustered with few misclassifications but returned four clusters, where one ALL-B sample was put into a singleton.

6.4.3. Case Study III: Inducing a Protein Taxonomy using Protein Sequence Similarities

In the last case study we clustered a hand-curated collection of 232 protein sequences originating from the SCOP database [84]. The dataset contains proteins of the superfamily amidohydrolases [23] belonging to 29 protein families which is a subset of the Brown *et al.* dataset (refer to Subsection 2.6.1 on page 62 for more information on the dataset). We aimed to subdivide the proteins according to the family annotation. The pairwise similarities were calculated by an all-vs.-all BLAST run. Since the input dataset consists of pairwise similarities, we cannot apply K-Means to the data. We created one parameter optimization run, with the settings shown in Table 6.2. The best performer (highest F-measure) was Hierarchical Clustering (0.987), followed by TransClust (0.986), Markov Clustering (0.923), and Affinity Propagation (0.724). The best clustering of Hierarchical Clustering was achieved with $k = 25$. TransClust performed best with $t \approx 52 \pm 5$, and the corresponding clustering contained 28 clusters. Compared to 29 being the actual number of protein-families, these clusterings appear to identify the gold standard classification well. Markov Clustering separated the proteins into 20 clusters with $I \approx 2.2$. Affinity Propagation performed best with preference $p = 46$, and resulted in a clustering with 18 groups. In general, ClustEval

was able to identify parameter sets for all clustering methods (except for Affinity Propagation) such that the resulting clusterings agreed well with the protein-families given in the gold standard (F-measure above 0.9).

6.5. Conclusion

Due to the multitude of available clustering approaches, data standards, and evaluation principles, cluster analyses require a high degree of manual work and expert knowledge. This affects the practitioner, e.g., a biologist as well as computer scientists developing novel clustering methods to the same degree. Until now, every new clustering approach was only tested on some selected datasets and compared only against a selection of competing tools. Every author decides on slightly different protocols and tweaks for the evaluation resulting in the newly developed tool to perform superiorly. Often, the data handling is quite poorly described so that reproducing the results can be hard and time consuming. All these reasons render it nearly impossible to compare existing tools with each other and their feasibility for different tasks based on the original publications of the tools.

ClustEval tackles these obstacles to ease the burden of conducting a comparative, large-scale cluster analysis. Our framework is tailored to the needs of different target groups and supports them in their typical day to day decisions concerning cluster analyses:

Non-Expert The non-expert can inform himself on the website about the available clustering tools typically used in bioinformatics. Depending on his dataset, he can identify similar datasets included in the platform and is shown the performance of the different tools. After he has made the decision to use a certain tool, he can track down the program parameters the platform used to achieve the presented results. In short, the website consults the practitioner what tool to use for which dataset and also suggests good parameters.

Expert Experts can download the entire back end and define their own analysis runs. That includes all steps from creating artificial datasets with gold-standards, automated threshold probing and the execution of different tools on different datasets. All of that is supported with numerous analysis features and tools for the user.

Developers As the entire framework is open-source and was designed to be as flexible as possible, every user can extend the framework. Developers of new clustering tools can include their new approach and compare their tool against the others in a standardized and transparent manner. That ensures that all results were produced fair and bias free and can easily be reproduced by researchers all over the world.

Furthermore, we suggest standard input and output formats for clustering tools in order to ease the interoperability between different clustering approaches. With the

three case studies, we have demonstrated the effectiveness of ClustEval. Each study was set-up within minutes, and provides the user with conclusive, fair, and high-quality results.

Results of this Chapter

- ClustEval gives an overview of the confusing field of clustering algorithms and their usage.
- Experts can set-up and run complex cluster analyses in a highly automatized fashion including parameter training.
- Universal standard input and output formats were suggested helping to increase the compatibility between different approaches.
- Availability: `http://clusteval.mpi-inf.mpg.de`

7. Biological Applications

In the course of the thesis, we have discussed novel methods in order to tackle massive biological datasets. We have already demonstrated the feasibility of these methods for clustering approaches in general and have shown that including support for missing values drastically reduces the computational effort for the calculation of the similarities between the objects.

In this chapter, we present two real-world applications of a cluster analysis in systems biology. The first study uses clustering of homologous proteins in order to transfer conserved regulations of a model organism to several non-model organisms. In a second study, we identify the core-genome of 89 actinobacteria (the core-genome can be seen as the set of common house-keeping genes shared among the organisms of an entire phyla). We aimed at identifying pathogenicity specific genes among them. That case study also suggest a solution for the long-standing problem of finding a good density parameter in the case of homology detection without a gold standard.

7.1. EHECRegNet

This section is based on the publication [93]:

Richard Röttger[1], Josch Pauling[1], Andreas Neuner, Heladia Salgado, Julio Collado-Vides, Prabhav Kalaghatgi, Vasco Azevedo, Andreas Tauch, Alfred Pühler, Jan Baumbach. **On the trail of EHEC/EAEC - Unraveling the gene regulatory networks of human pathogenic *Escherichia coli* bacteria.** *Integrative Biology.* 2012 Jul;4(7):728-33. doi: 10.1039/c2ib00132b.

[1] Joint first authorship of Richard Röttger and Josch Pauling.

Objectives of this Section

- Combine TransClust/TransClustMV with several data sources in order to transfer conserved transcriptional regulations between organisms.
- Use these findings to identify essential regulatory pathways for EHEC's pathogenicity.

7.1.1. Introduction

In this case study, we utilize TransClust for identifying evolutionarily conserved gene regulations among several organisms. As already mentioned in the motivation of the thesis on page 13,even for the best researched organisms like *Escherichia coli* K-12, we estimated in a previous study that in the best case, only 37 percent of the transactional regulatory network is known [102]. This is due to the costs and effort required for identifying a gene regulation in the wet-lab. Thus, mechanisms are needed in order to transfer the knowledge acquired in those model organisms to other organisms of interest.

As an example, we focus in this section on pathogenic *Escherichia coli*, such as Enterohemorrhagic *E. coli* (EHEC) and Enteroaggregative *E. coli* (EAEC) which are globally widespread bacteria. They have gained significance as a serious public health problem since 1982 and are causing sporadic outbreaks all over the world with up to 75,000 annual infections in the United States alone [63]. Some strains may cause the hemolytic uremic syndrome (HUS), which approximately 10% of the patients with an EHEC infection develop. Typical symptoms of HUS include haemolytic anaemia, thrombocytopenia and acute renal failure. It can cause neurological complications in 25% of the patients and chronic renal sequelae in 50% of survivors. In 3-5% HUS is fatal. Between 63% and 85% of EHEC infections derive from the consumption of contaminated food[2]. The cause for pathogenicity was attributed to five O157 strains, also referred to as the "gang of five", mostly present in raw meat, milk and eggs[3]. In fall 2011, we observed an epidemic outbreak of a new serotype in Western Europe, mainly in Germany[4]. In contrast to previous outbreaks the rare serotype O104:H4 was isolated and identified as the cause and it was most probably transferred to humans via contaminated sprouts[5]. It is yet unknown if the sprouts' seeds

[2]Enterohaemorrhagic *Escherichia coli* (EHEC): `http://www.who.int/mediacentre/factsheets/fs125/en/` Accessed on 22.07.2013

[3]German Scientists Finger Rare Serotype in Massive *E. coli* Outbreak: `http://news.sciencemag.org/scienceinsider/2011/05/german-scientists-finger-rare.html` Accessed on 22.07.2013.

[4]Phage on the rampage: `http://www.nature.com/news/2011/110609/full/news.2011.360.html` Accessed on 22.07.2013.

[5]EHEC-Ausbruch: BfR bestätigt Kontamination von Sprossen mit O104:H4 `http://www.bfr.bund.de/de/presseinformation/2011/17/ehec_ausbruch__bfr_bestaetigt_kontamination_von_sprossen_mit_o104_h4-70934.html` Accessed on 22.07.2013.

were already contaminated or if there is the possibility of a direct smear-infection. The complex of major virulence factors of O104:H4 includes the production of Shiga toxin, bacterial adhesion, flagellar motility, tellurite dioxide insensitivity and antibiotic resistances. Shiga toxins are able to cross the intestinal barrier of the host and bind to the endothelial cells, inhibiting protein synthesis or inducing apoptosis. Shiga toxin virtually shuts down the protein machinery of the susceptible cell [65]. All known EHEC virulence determinants are located on mobile genetic elements, often horizontally acquired leading to a dynamic evolution of EHEC, including various genetic mechanisms. The major virulent genetic compound is the locus of the enterocyte effacement (LEE) pathogenicity island. The key regulator, Ler, is essential for the expression of the LEE operon genes, including those encoding a type III secretion pathway as well as effector proteins and bacterial adhesin. LEE genes are also regulated by a mechanism of cell to cell signaling (quorum sensing) involving the production of hormone-like compounds, activating genes such as *qseB/C*. Flagellar motility, encoded by the *flhD/C* contributes to colonization and adherence to epithelial cells during an EHEC infection [79]. Figure 7.1 illustrates the molecular genetic program that is involved in EHEC's pathogenicity and toxicity. Because the O104:H4 has acquired both enteroaggregative and enterohemorrhagic characteristics it has recently been suggested to denominate it as an EAH (entero-aggregative-hemorrhagic) *E. coli* [24].

To contribute to the elucidation of gene regulatory interactions in pathogenic *E. coli*, we developed the integrated online database and analysis platform EhecRegNet. We transferred known regulations from *E. coli* K-12 to 16 human pathogens (see Table 7.1 for a summary). These virulent strains were sequenced, annotated and deposited with NCBI [19]. The platform is based on the identification of evolutionary conserved regulatory DNA sequences of the *E. coli* K-12 and the pathogens. This large-scale approach to the reconstruction and characterization of the pathogens' genetic control mechanism networks allows for a better understanding of their survival strategies and supports the development of new treatments.

EhecRegNet is a publicly available data source and interactive analysis platform for the transcriptional gene regulatory interaction networks of human pathogenic *E. coli* bacteria. It consists of three parts:

1. Integrated interaction data from the harmless *E. coli* K-12 strain and interactions transferred to the pathogens by identifying conserved regulatory DNA sites.

2. Access to the reconstructed networks as well as network visualization and analysis facilities available in the web.

3. In addition, we provide web-based access to further prediction methods as well as methods for the analysis of user-uploaded data, for instance gene expression data, together with stored network data.

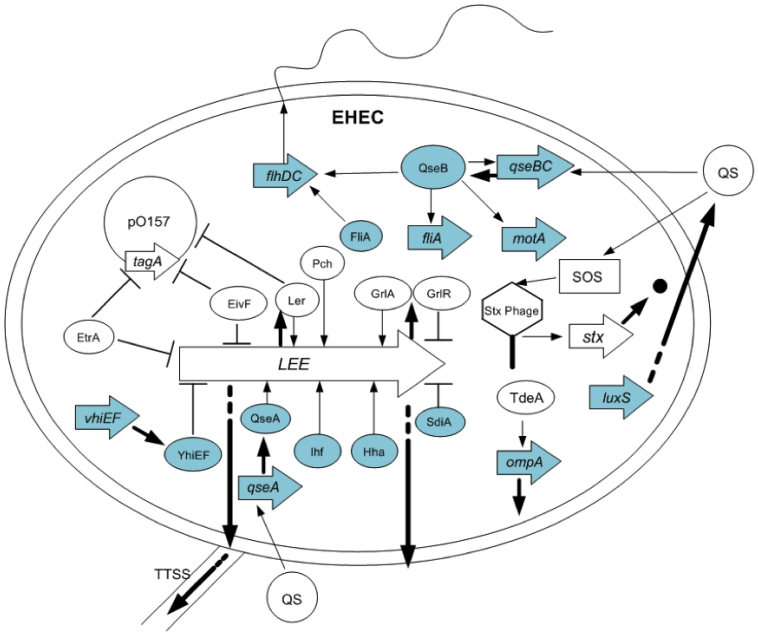


Figure 7.1.: This figure depicts the molecular genetic program of EHEC's pathogenicity and toxicity. Illustrated is the transcriptional gene regulatory interplay of those genes and proteins that are associated with EHEC's pathogenicity or toxicity capabilities (see text). Elements colored in blue are conserved in *E. coli* K-12. Abbreviations and symbols: Arrows = genes, circles = proteins, QS = quorum sensing, TTSS = type III secretion system

7.1.2. Materials & Methods

The main target of this platform is the transfer of known regulations of the model organisms (or source organism) to the human pathogenic *E. coli* strands (the target organisms). Figure 7.2 gives a brief overview of this process. Our main strategy follows suggestions from [13]. A transcriptional gene regulation can be described as follows: (a) certain gene products, here proteins, can act as so-called transcription factors (TF). (b) Such a transcription factor has the ability to bind to the DNA on those positions which encode the transcription factor binding site (BS). (c) Whenever a gene has this binding site in its upstream sequence and the TF binds to that binding site, the transcription factor may regulate the expression of that gene (which is called a target gene (TG)). Regulating means that the transcription factor can either increase the expression of this target gene or inhibit the expression. A transcription factor itself can also be the target of a regulation. We require three conditions to be met for transferring a known regulation into a new organism:

1. The transcription factor is conserved
2. The target gene is conserved
3. The binding site of the regulator is found in the upstream region of the target gene

Organism	Genes	RIs	TFs	Reg. genes	TFBSs
E. coli 536 (NC_008253)	4619	2206	116	978	1054
E. coli 55989 (NC_011748)	4759	2311	126	1014	1109
E. coli E24377A (NC_009801)	4749	2310	124	981	1098
E. coli IAI39 (NC_011750)	4730	2164	119	981	1081
E. coli O103:H2 str. 12009 (NC_013353)	5050	2345	123	1011	1119
E. coli O111:H- str. 11128 (NC_013364)	4968	2264	126	1001	1078
E. coli O127:H6 str. E2348/69 (NC_011601)	4549	2095	116	969	988
E. coli O157:H7 str. EC4115 (NC_011353)	5315	2337	121	1006	1074
E. coli O157:H7 str. EDL933 (NC_002655)	5297	2594	121	1097	1071
E. coli O157:H7 str. Sakai (NC_002695)	5229	2458	121	1036	1083
E. coli O157:H7 str. TW14359 (NC_013008)	5255	2485	119	1046	1066
E. coli O26:H11 str. 11368 (NC_013361)	5361	2368	122	1028	1117
E. coli O55:H7 str. CB9615 (NC_013941)	5014	2376	110	1112	557
E. coli S88 (NC_011742)	4692	2304	116	1006	1101
E. coli str. K-12 substr. MG1655 (NC_000913)	4319	3489	174	1440	2075
E. coli UMN026 (NC_011751)	4823	2475	127	1058	1128
E. coli UTI89 (NC_007946)	5017	2332	116	1001	1090

Table 7.1.: This table summarizes the database content of the EhecRegNet platform. The data for *E. coli* K-12 was integrated from the RegulonDB database and subsequently transferred to the pathogenic *E. coli* bacteria. On average 68% of the known *E. coli* K-12 gene regulatory interactions are conserved in the pathogens. Abbreviations: TFs = Transcription factors, RIs = Regulatory interactions, Reg. genes = Regulated target genes, TFBSs = Transcription factor binding sites

Only if all these conditions are fulfilled by the target organism, we consider that regulation as conserved. For the first two tasks, we require the knowledge of homologous proteins of the source and target organisms. As similarity function, the BeH score of an all-vs.-all BLAST run was calculated on the protein sequences of all organisms. Here, we used an E-Value cut-off of 0.01. Refer to Section 2.2.3 on page 29 for further details of using BLAST as a similarity function. Following the suggestion from [123], we used a similarity threshold of 40 for TransClust, which corresponds to an average intra-cluster E-value of 10^{-40}. The resulting clusters are interpreted as the set of homologous proteins.

In order to fulfill the third condition, the binding sites of the TFs in the model organism must be found in the target organism in the upstream sequence of the target gene. Therefore, we learned so-called position specific scoring matrices from known binding sites of each conserved TF and subsequently applied PoSSuMsearch [16] with a p-value cut-off of 10^{-4} to the upstream sequences of the orthologous target genes ($-460 \ldots +20$ bp relative to the gene start).

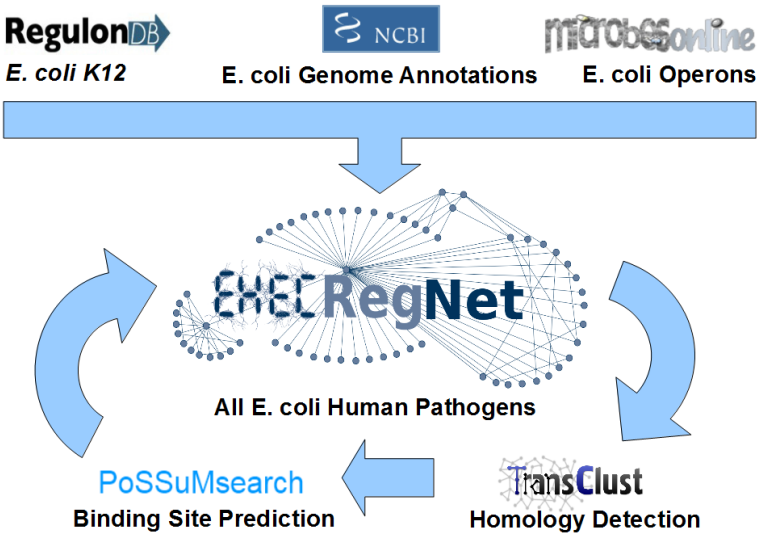


Figure 7.2.: The figure depicts the inter-species network transfer implemented in the back end of the EhecRegNet system. The experimental version of EhecRegNet consists of gene annotations obtained from NCBI, operon predictions downloaded from MicrobesOnline, and experimentally verified transcriptional gene regulatory interactions for *E. coli* K-12 from RegulonDB. For the predicted version, we first detect homologous genes by utilizing BLAST and Transitivity Clustering. Second, we identify putative transcription factor (TF) binding sites (TFBSs) for homologous TFs in the upstream sequences of homologous target genes by means of the PoSSuMsearch software. Whenever TF, target gene and TFBS are conserved between *E. coli* K-12 and the pathogens, we consider the regulatory interaction as conserved (see text for details). All conserved interactions are added to the predicted version of EhecRegNet; see Table 7.1 for a database summary.

Whenever we meet all three of the above described conditions, the EhecRegNet data integration pipeline considers a gene regulatory interaction as conserved and adds the corresponding regulation to the EhecRegNet database. Furthermore, if the target gene is the first gene of an operon, the entire operon is considered as being regulated. The same prediction model was already used in [14, 13, 15], where further details on the used parameters and their effect on the prediction accuracy are given. We set all parameters in such a way that our predictions are reliable, but not necessarily comprehensive (i.e., minimizing the false-positive rate).

In order to perform the described steps, EhecRegNet requires incorporating knowledge from several different sources. First, we integrated the genome annotations of 16 pathogenic *E. coli* species (including plasmids) as well as the *E. coli* K-12 genome annotation from the NCBI database. We also included operon prediction data with EhecRegNet provided by MicrobesOnline [33]. Note that in the case of *E. coli* O55:H7 str. CB9615 we used data from OperonDB [96]. Afterwards, we inte-

grated 3,489 known regulatory interactions from the RegulonDB database [50]. Its' content covers experimentally proven regulations for *E. coli* K-12, the best studied organism on Earth [13, 50, 68]. Here, we excluded all regulations marked as obsolete and those with unknown regulatory effect (repression or activation). In addition, we preprocess all transcription factor binding sites with MoRAine [124]. This is a sequence motif re-adjustment tool that enhances the information content of a sequence motif in order to improve profile-based binding site scans; refer to [124] for details. The EhecRegNet data integration pipeline performs all these steps in an automated manner. This allows us to easily include more genome annotations for more *E. coli* pathogens should they become available in the future. Further information regarding all the genes, functional annotations, regulatory pathways, etc. may be found at the gene detail pages of the EhecRegNet web sites.

7.1.3. Results & Discussion

EHECRegNet provides a reliable and comprehensive source for transcriptional gene regulation of *E. coli* K-12 and 16 pathogenic EHEC strands. Nevertheless, the database content is not static and will be extended in the future. New gene annotations, updated RegulonDB content or the outbreak of new strands require a rerun of the entire transfer pipeline. This task usually takes a couple of days to weeks and with every additionally included EHEC strand the effort of completing the transfer will even increase. Particularly when facing a new outbreak, a fast response time is crucial. Our newly developed clustering approaches, TransClustMV and ActiveTransClust, reduce the time for updating the database content and running the transfer pipeline to mere hours instead of days or weeks.

We illustrate the power of EhecRegNet with nine genes associated with pathogenicity or toxicity and evolutionarily conserved in the *E. coli* K-12 strain: *qseB*, *flhD*, *flhB*, *fliA*, *motA*, *ompA*, *ompR*, *yhiE*, and *yhiF*. As for many others, we transferred transcriptional regulatory interactions for these nine genes from *E. coli* K-12 to the pathogens. While some of the identified TFs regulating the nine genes are global hubs, others are more specific, involved in the regulation of a comparably small number of genes. We suggest these specific regulatory interactions as potential targets for further wet-lab validation. Regarding *yhiE* (*gadE* in *E. coli* K-12), we found five TFs. Two of them are predicted to regulate *yhiE* in all 16 pathogens. The *yhiE*-regulation of the three others is conserved in 15 of the 16 pathogens. The *gadW* and *gadX* TFs are interesting because they regulate a small number of genes, 10 and 22 respectively, while *crp* is regulating 386 genes. In the case of *motA*, we have found 8 TFs in 15 pathogens, with regulators acting on 4 genes (*irhA*), 12 genes (*ompR*), 28 genes (*rcsA*/*B*), and the key regulator *crp*. For *fliA*, we found three TFs occurring in at least 14 of 16 pathogens, regulating 79 genes (*flhD*/*C*) and 82 genes (*nsrR*). We identified several TFs regulating *flhB*, but no consistent picture for all or most of the pathogens. However, we detected potential regulations of *flhB* by *flhD* in 15 strains and through *flhC* in 14 pathogenic strains. A putatively conserved regulatory interaction is more likely to be a true positive when we observe it in many pathogens.

Table B.1 on page 189 in the appendix displays a brief overview of the analyzed genes and the findings for each species separately.

Evaluation of these results is difficult since no gold standard exists. We are far away from having a set of closely related organisms with enough experimentally proven transcriptional interactions that would allow us to evaluate our approach on a large scale. However, a recent study [13] suggests that our strategy works well for four corynebacteria, where at least some TFs have been characterized for more than one organism. Consequently, since the *E. coli* pathogens are evolutionarily close to *E. coli* K-12, we believe that the EhecRegNet inter-species network transfer pipeline is appropriate for our *E. coli* organisms as well. Regarding our concrete results, note that EhecRegNet detects the known self-regulations of *qseB* in all pathogens but no other regulatory interaction. Furthermore, note that we also find all *fhlCD* self-regulatory interactions (B.1 on page 189).

To ensure reliable results that provide high-potential candidates for wet-lab studies the automatic inter-species transfer pipeline of EhecRegNet is comparably restrictive. In the examples discussed above, for all genes and for each of the pathogenic *E. coli* strains the transcription factor, the target gene and the binding site had to be conserved in order to define a predicted interaction. However, EhecRegNet's website offers access to several bioinformatics tools, one of them is the TFBScan feature. With TFBScan the user can semi-automatically detect transcription factor binding sites in the upstream sequence of a specific gene of interest. For Ler, one of the key TFs regulating the critical LEE operons in EHEC, we found two regulating transcriptional factors, *fruR* and *nagC*, using this feature. For one of the other main factors of virulence, the Shiga Toxine genes *stx1A/B*, organized in an operon, we found five potential binding sites of high confidence, delivering interesting new targets for further wet-lab validation.

7.1.4. Conclusion

EhecRegNet is the largest compilation of data about transcriptional regulatory interactions of *E. coli* pathogens publicly available. Our inter-species network transfer pipeline identified in average 68% of the *E. coli* K-12 regulatory interactions as conserved in the 16 human pathogenic *E. coli* bacteria (Table 7.1 on page 131). If interactions are not transferable, for in-orthologous genes, for instance, EhecRegNet provides integrated methods for detecting putative regulatory sites. EhecRegNet will support wet-lab researchers with reconstructing and characterizing the pathogens' genetic control mechanism network. It will allow for a better understanding of their pathogenicity and toxicity. This will support the development of urgently needed new treatments.

Results of this Section

- Almost 60% of the known regulations of *E. coli* K-12 were successfully transfered.
- Several transcriptional regulators of EHEC's pathogenicity pathway were unraveled.
- Availability: `http://www.ehecregnet.de` or `http://www.ehecreg.net`

7.2. Actinobacterial Core Genome

This section is based on the publication [100]:

Richard Röttger, Prabhav Kalaghatgi, Peng Sun, Siomar de Castro Soares, Vasco Azevedo, Tobias Wittkop, Jan Baumbach. **Density parameter estimation for finding clusters of homologous proteins - tracing actinobacterial pathogenicity lifestyles.** *Bioinformatics* (2013), 29 (2), 215-222, DOI:10.1093/bioinformatics/ bts653

Objectives of this Section

- Unravel the actinobacterial core gnome for tracing pathogenicity lifestyles.
- Development of a reliable method for finding a reasonable threshold in the absence a gold standard.

7.2.1. Introduction

In the last section, clustering was used in a prediction pipeline in order to unravel regulatory mechanisms in an organism. This approach has proven very useful not only for EHEC but also for *corynebacteria* [94]. In these approaches, the threshold was chosen relatively low, i.e., forming fewer bigger clusters. This poses no problem, as protein homology is not the only criterion for the prediction. Potential false positives (i.e., genes considered falsely as homolgous) get filtered out of the list of conserved regulations as the corresponding binding site cannot be found in the upstream region of this gene. Nevertheless, in most situations, picking a good threshold or parameter set is vital for the result quality.

Clusters of homologous proteins across a number of organisms allow for studying lifestyle-specific genetic repertoires, i.e., the genes that have homologous counterparts in all organisms or in a specific set of organisms. Such studies can lead, for instance, to the discovery of mutual proteins shared only among pathogenic strains of a certain phyla, thus suggesting new drug targets and wet laboratory candidates for vaccine design. The quality of such studies is highly dependent on the quality of the clustering process and consequently dependent on the choice of the clustering method and a good estimate of the parameter set.

Despite the efforts put into the development of new clustering algorithms, the general problem of detecting a good parameter set has widely been neglected. A clustering tool cannot 'know' *a priori* if we seek to find protein families (restrictive parameters) or protein superfamilies (weak parameters), for instance. It is intuitively apparent that choosing a very low threshold leads to many false-positives whereas a high threshold leads to many false-negatives.

Normally, such a density parameter is estimated by utilizing a gold standard by picking that threshold yielding the best agreement. In this section we present a robust method for selecting a suitable density parameter for the task of protein homology detection. We utilize TransClust but the method is suitable for any clustering tool. Using all protein sequences from the given set of organisms, we build our method upon two assumptions: (i) clusters of size equal to the number of input organisms are likely to contain housekeeping genes and thus should be over-represented, and (ii) clusters greater than the number of input organisms are more likely to contain many false positives (non-homologous genes). Maximizing (i) while minimizing (ii) allows us to estimate a meaningful threshold for discovering clusters of homologous proteins without manually curated gold standard associations for any of the proteins.

As an example for this novel method, we compute and analyze the core genome of 89 *actinobacteria*. The core genome is the set of evolutionarily highly conserved genes which are present in all organisms in question. The fact that these genes are conserved within all organisms suggests that the genes are of vital importance. We further divide them into four different groups of pathogenicity: non-pathogens (NPs), human pathogens (HPs), animal pathogens (APs) and opportunistic pathogens (OPs). See Table C.1 on page 211 in the Appendix for a complete list of the used organisms. We then study the class-specific genetic repertoire of the 89 actinobacteria. Refer to Subsection 2.6.2 on page 62 for a detailed description of this dataset.

There have been several studies about the actinobacterial evolution (refer to Gao and Gupta [51]). Most of them concentrated on phylogenetic tree reconstruction solely based on the DNA sequence information of the 16S RNA. Despite the many advantages of this method, it cannot provide insights into the evolutionary relationship on a species level [107]. Gao and Gupta in [52] used only a limited dataset of only a few genes that were expected to be conserved along the phylum for phylogenetic tree reconstruction. In several recent studies, best bidirectional hits from genome-wide all-vs.-all BLAST results of all genes were used for homology detection (Karberg *et al.* [64] or [53], for instance). This strategy, however, neglects the impact of careful BLAST cutoff evaluation, as well as the effect of transitive dependencies in the similarity function. Gene A may be similar to gene B, which is similar to gene C, but gene C is not similar to gene A. These problem instances can be 'repaired' with clustering tools such as TransClust. However, the problem of finding a reasonable density parameter remains with TransClust, as well as with any other clustering method.

7.2.2. Materials and Methods

7.2.2.1. Threshold Estimation

To reasonably investigate the clustering results, the density parameter must be set correctly such that most of the clusters actually contain groups of homologous proteins. In our study with >300,000 proteins from 89 different bacteria, we do not have a given gold standard that would allow us to find a reliable threshold. We will present

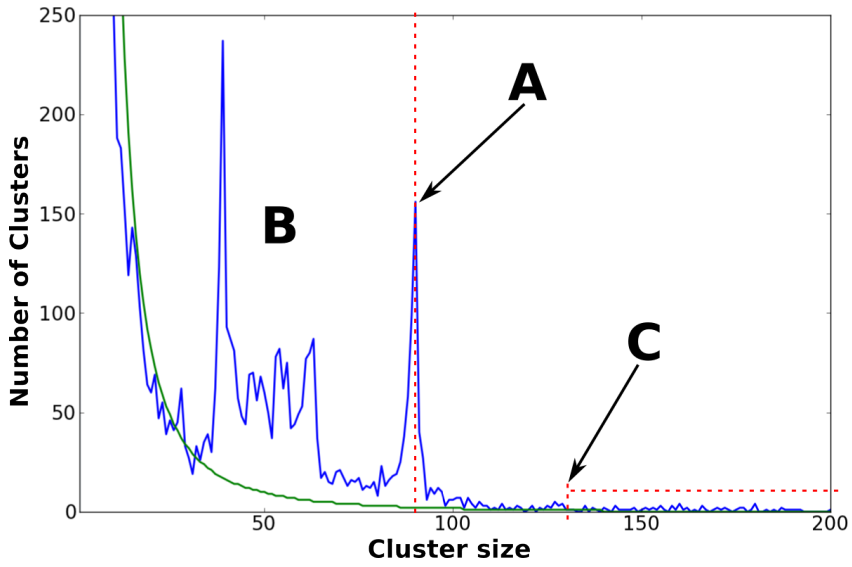


Figure 7.3.: Cluster size distribution of the 89 actinobacteria for similarity threshold 48 (which corresponds to a BLAST E-value cutoff of 10^{-48}). Arrow (A) highlights the core genome peak at cluster size 89. These peaks in area (B) represent more specific core genomes, for example, all mutual proteins of the different mycobacteria/corynebacteria strains. The beginning of the unspecific clusters is marked by arrow (C).

an approach that only uses intrinsic indirect information of the dataset to determine such a threshold.

In what follows, n denotes the number of species. This number, i.e., $n = 89$ in our study, is constant and independent on the chosen threshold. Our first assumption is based on the expectation of observing significantly more clusters of size n than clusters of other sizes, as housekeeping genes and essential genes are expected to be conserved across all bacteria. Thus, they are more likely to cluster together in a group of exactly (or almost exactly) 89 proteins. In our analysis, we observed a peak in the cluster size distribution (Figure 7.3) at $n = 89$, with most of these clusters containing exactly one protein from each of the 89 organisms. This gives evidence in favor of our first assumption. Setting the clustering threshold such that we maximize the size (height) of this peak would increase the number of allowed housekeeping gene. However, on the other hand, we cannot assess the number of false positives in these clusters directly, as we do not have a given reliable gold standard. What we require is a second measure for allowing us to minimize these false positives. Here, our second assumption is used: clusters with larger sizes (far bigger than n proteins) are likely to contain non-homologous proteins (i.e., false positives). The more a cluster size exceeds n, the more unlikely it is that this increase can be explained by true-positive paralogous proteins. We will use this assumption to obtain a measure for handling the number of false positives.

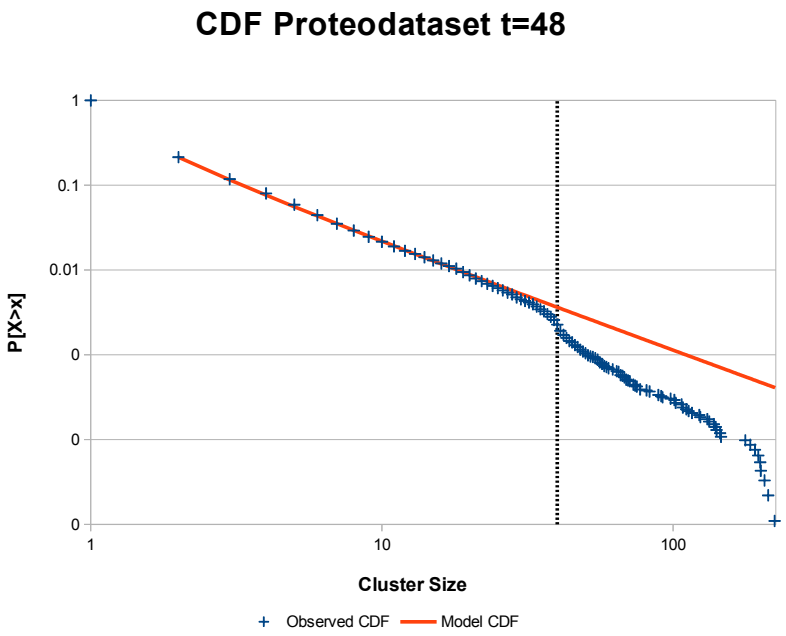


Figure 7.4.: The cumulative distribution function (CDF) of the cluster size distribution. The CDF denotes the probability to sample a value from a probability function X larger than x, i.e., $CDF(x) = \mathrm{P}[X > x]$. The picture displays the observed CDF when clustering the Proteo dataset (refer to Subsection 7.2.2.2 for more information on this dataset; we chose this dataset as it contains the greatest diversity of species resulting in the smallest relative core genome peak, thus the power-law is least disturbed compared to the other datasets) with a threshold of 48. The orange line indicates the ideal power-law following the parameters estimated for the observed distribution. We can see, that the cluster size distribution follows the power-law very well, at least till the core genome peak (in this dataset $n = 40$, indicated by the black dashed line).

Put in other words, our strategy is to vary the similarity threshold such that our TransClust-based clustering results yields the following two optimizations:

1. Maximize the number of clusters of size n (most likely containing common housekeeping genes).

2. Minimize number of large clusters (most likely containing many false positives).

To account for the first point, we have to separate the desired peak from the background distribution to get the relative peak height. The base line of the cluster size distribution seems to follow a power law. For that reason, we learned the best fitting discrete power-law:

$$P_{\alpha,x_{\min}}(x) = \frac{x^{-\alpha}}{\zeta(\alpha, x_{\min})},$$

with $\zeta(\alpha, x_{\min})$ being Riemann's Zeta function for the background distribution using the Python tools provided by Clauset *et al.* [31]. Figure 7.3 depicts the cluster size

distribution and the best fit power law for threshold 48 (we will later explain why we picked 48). We decided to approximate the background distribution with a power-law by visual verification (compare to Figure 7.3 and 7.4). The formal plausibility check if the power-law is a valid assumption (as suggested in [31]) can not be applied, as the peak of the core genome in fact distorts the power-law behavior of the cluster size distribution. Furthermore, as the influence of the background distribution on the relative peak height is rather small (again, compare to Figure 7.3), the optical verification serves our purpose at this point.

Let $\hat{\alpha}_t$ and $\hat{x}_{min,t}$ be the approximated parameters for the best fitting power-law for the cluster size distribution $D_t(x)$ for threshold t. The function $D_t(x)$ gives the absolute number of clusters of size x. Furthermore, m_t denotes the number of observations, i.e., the total number of clusters, again for threshold t. We now define the relative peak height $h_t(x)$ as

$$h_t(x) = D_t(x) - P_{\hat{\alpha}_t,\hat{x}_{min,t}}(x) \cdot m_t$$

where $P_{\hat{\alpha}_t,\hat{x}_{min,t}}(x) \cdot m_t$ denotes the expected number of observation of a perfect power-law of the given sample size m_t, as $P_{\hat{\alpha}_t,\hat{x}_{min,t}}(x)$ is a probability function. In the following, we refer to the relative core-genome height $h_t(n)$ as h_t.

In order to address the second optimization criterion, we need to penalize the occurrence of unrealistically large clusters (false positives). In this work, we define such a cluster as a cluster containing more than $1.5 \cdot n$ proteins. It is very unlikely, that there are clusters of that size containing only real homologous and functional identical proteins, because our actinobacterial dataset is quite diverse. A "real" cluster of size $\frac{3}{2} \cdot n$ would imply that at least half of the species must have undergone the same duplication event. That means this duplication event most likely happened at an evolutionarily very early time point in their common ancestor. On the other hand, the genetic variation was small enough such that these paralogous proteins still belong to the same cluster of homologous proteins. If that would happen to be a common case, one would also expect core-genome peaks for paralogous proteins, e.g., at $2 \cdot n$ or $3 \cdot n$. We were not able to identify such a peak for any of the similarity thresholds. In conclusion a cut-off for unrealistic big cluster at $1.5 \cdot n$ is reasonable and the accidental punishment of real paralogous clusters is negligible. Thus we define the number of unrealistic clusters u_t for threshold t as

$$u_t = \sum_{x > \frac{3}{2} \cdot n} D_t(x).$$

In a final step, we have to combine both quality measures to a single overall quality value that we can assign to the TransClust results for the varying thresholds. As h_t

and u_t are two completely different measures, we scale them to the range $[0, 1]$ with T being the set of all used thresholds:

$$\begin{aligned} h'_t &= \frac{h_t - \min(h_i, \forall i \in T)}{\max(h_i, \forall i \in T) - \min(h_i, \forall i \in T)} \\ u'_t &= \frac{u_t - \min(u_i, \forall i \in T)}{\max(u_i, \forall i \in T) - \min(u_i, \forall i \in T)} \end{aligned}$$

Following the F-measure, we calculate our final quality measure $Q(t)$ as the harmonic mean of both of them:

$$Q(t) = 2 \cdot \frac{h'_t \cdot (1 - u'_t)}{h'_t + (1 - u'_t)}.$$

We are using $(1-u'_t)$ such that lower numbers of unrealistic clusters result in better quality measures. We may now use this approach to find that similarity threshold t of TransClust, which gives the best quality measure $Q(t)$. Figure 7.5 plots $Q(t)$ for several TransClust results for thresholds ranging from 8 to 100, corresponding to a BLAST E-value of 10^{-8} to 10^{-100}.

7.2.2.2. Robustness Analysis

So far, we have derived a quality measure $Q(t)$ using only intrinsic information of the provided dataset. On the other hand, actinobacteria are known to be quite diverse, suggesting that our dataset is biased. For example, we have 35 different strains of *Mycobacterium tuberculosis* which are all likely to be more similar to each other than to the other actinobacteria. In order to respect for this potential bias, we split our datasets to investigate the stability of our approach. The following datasets were created:

- Myco-Only: All organisms of the genus mycobacteria (here: 55 species).
- Coryne-Only: Same as Myco-Only but with all corynebacteria (here: 27 species).
- Rand-20: Here we randomly selected 20 out of the 89 species without replacement. We created 20 such datasets, in order to get an impression of the variability of our approach.

As expected, our approach is limited by the level of biological diversity amongst the studied organisms. Although the actinobacerial phylum already is quite diverse, we also selected a dataset consisting of 40 different proteobacteria. Proteobacteria comprise one of the largest bacterial phyla with a large genetic diversity [108]. In the remainder of this section we will call this the Proteo dataset. With Proteo we aim to assess the stability and the limits of our approach for more diverse genomes. We used the protein sequences of ten bacteria of each of the following four proteobacterial subgroups: alphaproteobacteria, betaproteobacteria, gammaproteobacteria and the delta/epsilon subdivisions (40 genomes in total). Refer to Table C.2 on page 213 in the appendix for a detailed description. As we only use intrinsic information 'hidden'

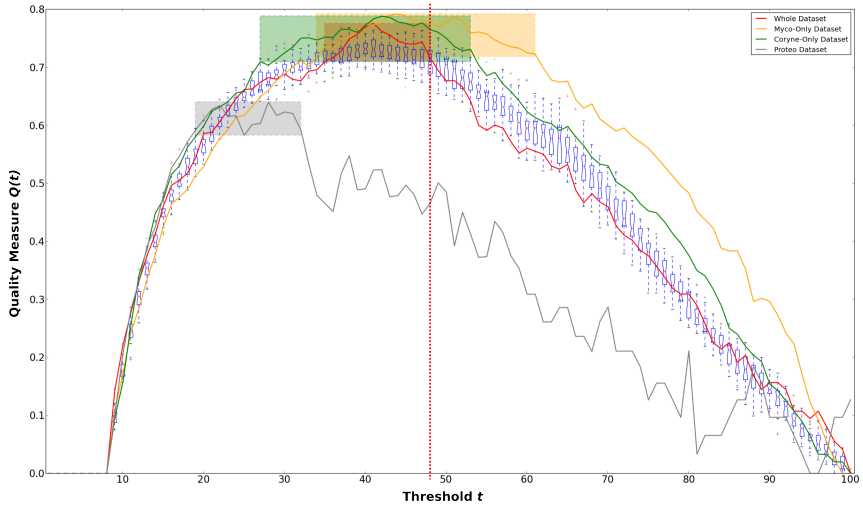


Figure 7.5.: In this figure, we plot our quality measure against the similarity threshold of TransClust. The red plot represents the quality measure $Q(t)$ for the entire dataset and the blue box-and-whisker plot in the background represents the variance and mean of all Rand-20 datasets (see text). The green and orange lines plot the quality measure for the Coryne-Only and the Myco-Only dataset respectively (see text). The three boxes in the plot mark the pick range, i.e., that range of thresholds where we see 10% of the best quality hits $Q(t)$. For the two rather phylum-biased datasets, i.e.Myco-Only and Coryne-Only, the pick range is larger than the pick range of the entire dataset. Notably, the pick range for the entire dataset is completely contained in the pick range of both, the Myco-Only and the Coryne-Only datasets. The gray line indicates the quality measure for the Proteo dataset. This dataset is too diverse for the presented quality measure, which is indicated by the generally lower quality values and the shifted box toward a weak threshold. The dotted red line indicates the threshold 48, which was chosen for the core genome analysis (see text).

in the dataset, we rely on a certain level of homogeneity amongst the genomes in order to receive a reasonably large 'core-genome peak'. Hence, we may expect a slightly lower quality measures for the more diverse Proteo dataset highlighting the limits of our approach.

7.2.2.3. The Actinobacterial Phylogenetic Tree

Given a meaningful clustering of homologous proteins, we may now calculate an inter-species similarity. Let $O = \{o_1, \ldots, o_n\}$ be the set of n organisms with $o_i = \{p_{i1}, \ldots, p_{in_i}\}$ as a set of n_i different proteins. Furthermore, let $C = \{c_1, \ldots, c_m\}$ be the set of m clusters. We define $\delta_{o_i}(c_k)$ to be the number of proteins that organism o_i has in cluster c_k. The function

$$\delta_{o_i,o_j}(c_k) = \begin{cases} 0 & \text{if } \delta_{o_i}(c_k) = 0 \vee \delta_{o_j}(c_k) = 0 \\ \delta_{o_i}(c_k) + \delta_{o_j}(c_k) & \text{otherwise} \end{cases}$$

denotes the number of mutual proteins in cluster c_k of organisms o_i and o_j if both organisms are represented by at least one protein. The similarity function $s(o_i, o_j)$ between two organisms o_i and o_j is now defined as

$$s(o_i, o_j) = \frac{\sum_{c_i \in C} \delta_{o_i,o_j}(c_i)}{n_{o_i} + n_{o_j}}.$$

This is basically the number of all mutual proteins of o_i and o_j normalized by the total number of proteins of both species. This normalization is done to prevent a bias of the similarity towards species with larger genomes (more genes).

These inter-species similarities fulfill all properties for a similarity function required for TransClust. In order to create a phylogenetic tree, we ran TransClust in hierarchical mode with option "top-down". Set in hierarchical mode, TransClust starts with a very low threshold that is increased over several iterations. As result, in the first iteration we obtain one big cluster containing all species. With more restrictive thresholds, the cluster(s) are divided into smaller clusters until each species ends in its' own singelton cluster. The clustering result of all iterations is used to generate a phylogenetic tree. This tree is now based on the whole-genome repertoire of all actinobacteria. Note that we construct this (simple) tree for supporting our threshold estimation procedure, rather than introducing a new phylogenetic tree reconstruction methodology.

7.2.3. Results & Discussion

7.2.3.1. Threshold Estimation

First, we will discuss the evaluation of the threshold estimation method. Figure 7.5 illustrates the stability of our approach. In particular, the results of the 20 randomly sampled Rand-20 datasets are a very good indicator for the reliability of our approach (refer to Table 7.2).

We define a threshold pick-range $R_D = \{t_i, \ldots, t_k\}$ as the set of all thresholds, where the quality measure $Q(t)$ exceeds 90% of the best threshold of dataset D, i.e., that similarity threshold area where we find 10% of the best results. The pick-ranges for the different datasets are marked with a box in Figure 7.5. For the complete dataset, we observe a pick-range of $R_{\text{All}} = \{35, \ldots, 48\}$. In this range, the standard deviation of the 20 Rand-20 datasets is only about three percent from the mean.

As we expected, the Proteo dataset (gray line) shows a lower quality $Q(t)$ than the actinobacterial dataset(s). We also observe a left-shifted pick-range, i.e., towards a lower threshold, resulting in a less rigorous homology detection. The main reason for that is the smaller size of the proteobacterial core genome. This indicates that a single threshold for all species, ignoring the level or diversity, cannot sufficiently be detected and, for instance, the alphaproteobacteria should be investigated separately from the betaprotebacteria. But a deeper investigation of proteobacteria is outside the focus of our study.

	20×Rand-20			All Data	
t	$\mu(\mathbf{Q(t)})$	$\sigma(\mathbf{Q(t)})$	**Ratio**	$\mathbf{Q(t)}$	$\mathbf{\Delta}(\%)$
35	0.716	0.0152	2.13%	0.711	-0.72%
36	0.719	0.0196	2.73%	0.717	-0.28%
37	0.724	0.0195	2.69%	0.728	0.56%
38	0.729	0.0157	2.15%	0.749	2.87%
39	0.732	0.0186	2.54%	0.757	3.46%
40	0.731	0.0200	2.74%	0.771	5.36%
41	0.726	0.0192	2.64%	0.776	6.84%
42	0.723	0.0181	2.50%	0.763	5.44%
43	0.721	0.0215	2.98%	0.756	4.90%
44	0.719	0.0232	3.22%	0.746	3.67%
45	0.717	0.0232	3.23%	0.741	3.34%
46	0.715	0.0259	3.63%	0.740	3.49%
47	0.711	0.0257	3.61%	0.744	4.64%
48	0.706	0.0263	3.73%	0.717	1.62%

Table 7.2.: This table shows the exact values for the evaluation of the threshold estimation. The left part represents the results of the 20 Rand-20 datasets, showing the mean (column "$\mu(Q(t))$"), the standard deviation (column "$\sigma(Q(t))$") and the percentage of the standard deviation with respect to the mean (column "Ratio"). For comparison, the right part displays values for the entire dataset, subdivided into a column showing the quality measure ("$Q(t)$") and column "Δ" displays the percentage deviation of "$Q(t)$" from "$\mu(Q(t))$".

We now discuss the Myco-Only dataset. The quality measure is better than for the other datasets and the pick-range is larger (the range of suitable similarity thresholds is bigger). This is mainly contributed to 35 strains of *M. tuberculosis* in a dataset with a total of only 55 species. As the different strains of *M. tuberculosis* are very closely related, there is less variance in the clustering result with respect to the threshold. In other words, the proteins of the core-genome cluster together quite early (for weaker thresholds) and variance only occurs for the less similar proteins of the non-tuberculosis species. Therefor, the relative core-genome peak hight stays pretty stable for a broader range of thresholds compared to the entire dataset.

We suggest, that all thresholds from the pick-range are good candidates. We decided to choose the most restrictive one, i.e., 48, in order to further reduce the possibility of false-positives in the homology detection and thus enhance the confidence in the presented actinobacterial core-genome. We marked this threshold with a dashed line in Figure 7.5.

7.2.3.2. Pathogenicity as a Genetic Model

In this first application of our previously obtained clusters of homologous proteins, we study the relationship between the genetic repertoire and bacterial life styles,

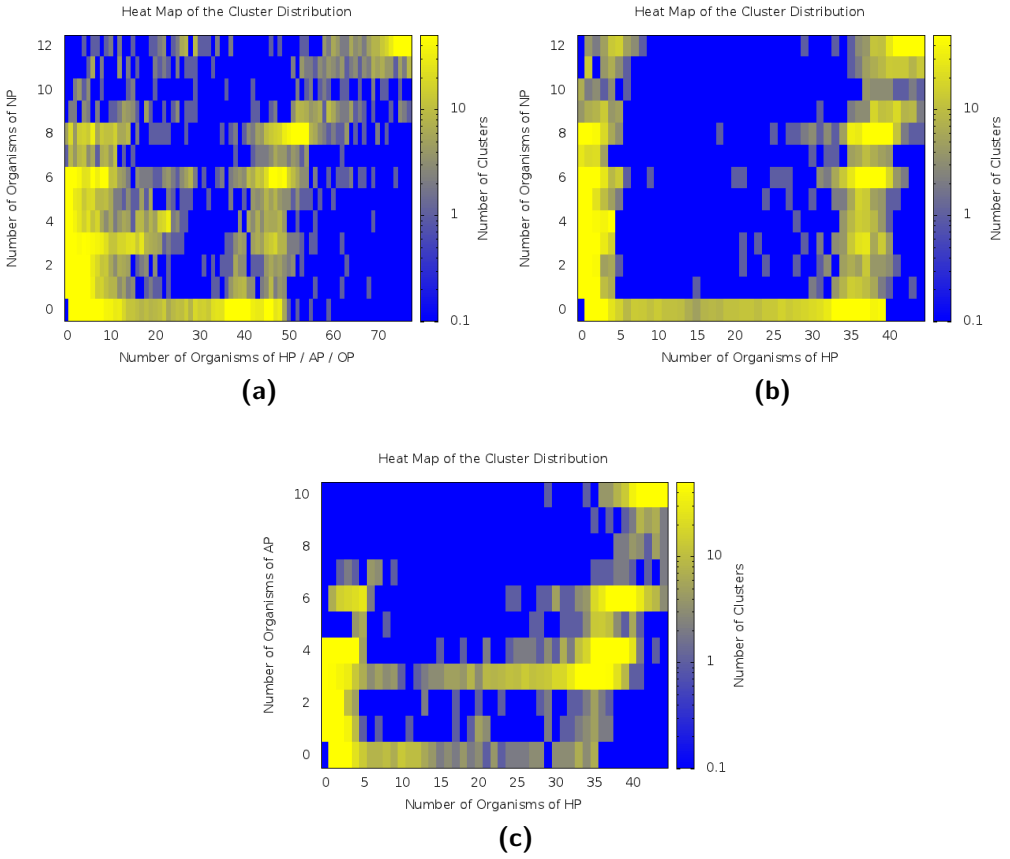


Figure 7.6.: (a) for example, represents on the x-axis all pathogens and on the y-axis only non-pathogens. The colors encode the number of clusters that contain x "x-axis-type-pathogenic" and y "y-axis-type-pathogenic" species. Note that we ignore paralogs and count every species only once. The core-genome can be found in the top right corner, whereas the top-left and bottom-right corners represent the exclusive core-genomes. There are no peaks in the latter two areas, which means that there are no proteins that uniquely distinguish between the two pathogenicity classes. Note the log-scale of the color range.

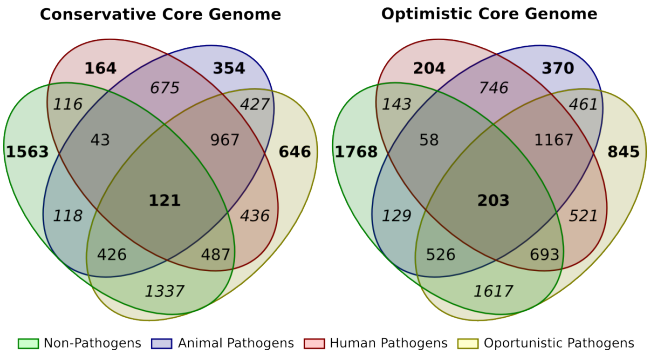


Figure 7.7.: These Venn diagrams depict the number of shared clusters in each possible intersection of the four different kinds of pathogenicity with at least three proteins (for conservative and optimistic; see text). These intersections are disjoint, for example the intersection of the non-pathogenic core-genome and the human pathogenic core-genome does not contain the human-pathogenic-only clusters. The core-genome contains all clusters, which contain proteins of all species. We marked the NP/HP/AP/OP-only clusters and the core-genome itself with bold font, for a intersection of two areas we used italic font. Refer to the text for a discussion of this figure.

pathogenicity classes in our case. In particular, we are looking for genes that we find exclusively in a certain class of species, pathogens, for instance. Most likely, those genes would be conserved across several different pathogenic phyla and thus build a cluster that contains no proteins from non-pathogen organism. In the following, we work with the TransClust clusters that we obtained by using the conservative threshold of 48, estimated as described above. In the following we will distinguish between four different types of pathogenicity:

- **HP**: human pathogens (44 bacteria)
- **AP**: animal pathogens (10 bacteria)
- **OP**: opportunistic pathogens (23 bacteria)
- **NP**: non-pathogens (12 bacteria)

Note: Opportunistic pathogens are generally not infectious but normally act commensal and do not harm the host. However, they can cause diseases in the case of a weak host's (immune) resistance. Figure 7.6 depicts distributions of the cluster size overlaps between different combinations of the pathogenicity classes. Furthermore, we provide datasets containing all specific core-genomes, the general core-genome and all possible combinations, for example clusters containing only proteins from HP and AP but not from OP and NP. These datasets are disjoint, e.g., the combined core-genome of HP and AP does not contain the only-HP and only-OP clusters.

Some clusters were bigger than the number of species. Hence, some species must contribute with two or more proteins. That can happen by means of gene duplication events or due to clustering mistakes, i.e., false-positives in the homology detection. Therefore, we provide the core-genome datasets in two different "flavors":

- **Optimistic**: All clusters with three or more proteins (includes paralogs).
- **Conservative**: Only those clusters from the Optimistic, where the number of proteins equals exactly the number of involved species (no paralogs).

The general core-genome includes only those clusters where all 89 species are involved. In the conservative case, these clusters are additionally limited to a size of exactly 89 proteins.

Figure 7.7 depicts a Venn diagram containing the number of clusters in the different categories. Of particular interest are "distinctive clusters" which lack participating organisms of at least one type of pathogenicity. Thus the core-genome and all other clusters containing proteins of species of NP, HP, AP and OP are not on that list, because they do not provide information on how to separate the different types of pathogenicity. One can clearly observe a connection between human and animal pathogens. 1,685 out of 2,888 HP and 3,010 AP distinctive clusters are shared, which account for more of half of the respective distinctive clusters in HP and AP. In contrast to this, only 646 distinctive clusters contain proteins from HP and NP and even less, 587, are conserved across AP and NP. The opportunistic pathogens seem to be less specific as they overlap very well distributed with all other categories (HP: 1,890, AP: 1,820, NP: 2,250). All these results are based on the conservative core-genome with TransClust threshold 48, although a similar tendency can be observed in the optimistic case (data not discussed here). Although our results not totally fulfill our hope of seeing 100% pathogenicity-class-specific proteins, our findings clearly indicate a certain genetic divergence between the pathogenicity life styles.

7.2.3.3. Quality of the Homology Detection

As already mentioned, there is no gold standard covering our bacteria. This poses problems for assessing the appropriateness of clustering methods for homology detection. Although slightly beyond the scope of this study, we like to discuss the agreement of our results with existing prediction-based homology repositories, EggNOG [97] and the Ortholog Matrix Project (OMA) [36], for instance. OMA has the largest number of common species with our study and was shown to perform superior on this task [35]. We hence compare against OMA. We mapped 118,000 proteins of 30 species (9 corynebacteria, 17 mycobacteria, one nocardia, and three rhodococcus) against our actinobacterial dataset by their IDs and their sequences. Refer to Table C.3 on page 215 in the appendix for a list of mapped proteins and species. Finally, we removed all unmapped proteins from both actinobacterial datasets.

In order to assess the agreement of both datasets, we utilized the F-measure. We varied the TransClust threshold and compared the results against OMA (see Figure 7.8). For the best threshold(s) the F-measure of 0.7 is quite good. Most notably, however, is the observation that the F-measure is best for thresholds almost exactly within the pick-range that was suggested by our method (between 35 and 48). As with this paper we particularly focus on detecting a meaningful threshold, i.e., density parameter, for clustering algorithms (rather than studying the performance of

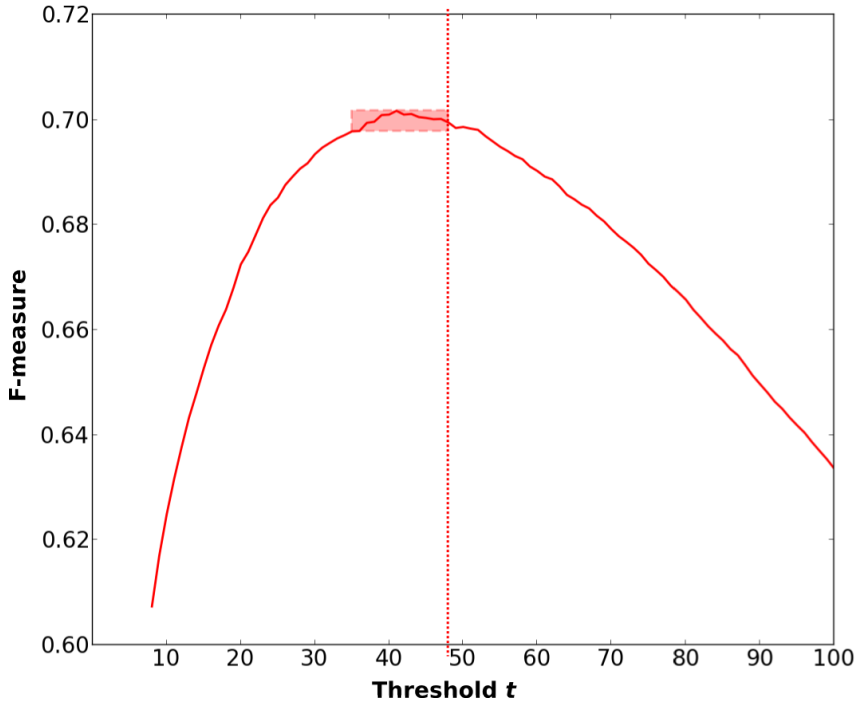


Figure 7.8.: Agreement with the OMA homology detection tool. Depicted is the development of the F-measure as a function of the clustering threshold. The red box marks the pick-range derived by using our model $Q(t)$. Remarkably, the best F-measures (agreements with OMA) are achieved for clustering results with thresholds in the pick-range we suggested using our method. The red dotted line indicates the threshold 48.

clustering algorithms for homology detection in general), this observation further strengthens our main conclusion. Furthermore, it would be hard to make a qualified statement about the quality of OMA compared to ours, as both methods are based on computer predictions.

7.2.3.4. The Actinobacterial Phylogenetic Tree

We utilized our above introduced inter-species similarity to perform a hierarchical clustering. With this, we were able to construct a phylogentic tree based on the whole-genome repertoire of all 89 actinobacteria. Figure 7.9 depicts the resulting tree. Whenever a cluster is split into sub-clusters, with increasing threshold, we branch in the tree accordingly. If a cluster sticks together for x decreasing thresholds, we set the length of the branch to $\log(x+1)$. This is necessary mainly for optical reasons, because some very closely related organisms stick together for many threshold which would result in very long branches. One can see that most of the mycobacteria cluster together while the other CMNR groups are slightly more separated. This observation is reasonable, given the different life styles and was previously reported

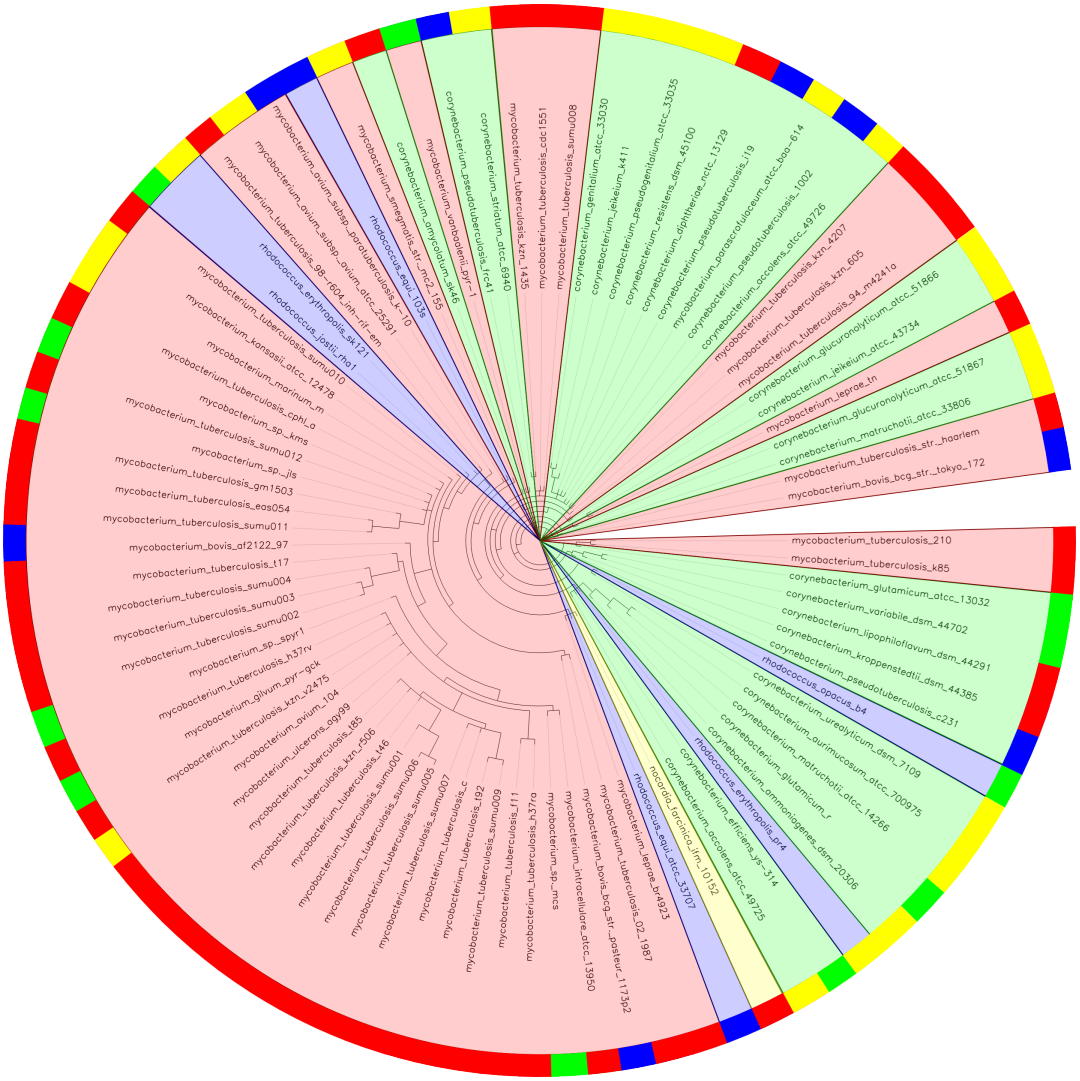


Figure 7.9.: Circular phylogentic tree based on the whole-genome repertoire of all 89 actinobacteria. The different subgroups of the CMNR group are marked with the following colors: mycobacteria, red; corynebacteria, green; rhodococcus, blue; nocardia, yellow. The colored ring on the outside idicates the pathogenicity: NP, green; HP, red; AP, blue; OP, yellow. The thick lines in the circle represent the tree, the thin lines are only for representational reasons.

in other studies, see the review from Ventura *et al.* [113], for instance. We emphasize that this tree is supposed to support our threshold estimation procedure, rather than introducing a new method for phylogenetic tree reconstruction.

7.2.4. Conclusion

To sum up, we studied the actinobacterial genetic repertoire with respect to four pathogenicity life styles. We used BLAST and TransClust for this purpose. Here, our main novel contribution was the estimation of a robust similarity threshold for TransClust. Therefore, we set the threshold such that we balance the size of the core-genome (number of clusters with exactly 89 genes/proteins; putative true positives) and the number of unreasonably larger clusters (putative false positives) based on the cluster size distribution. We studied the robustness of our method by using random sampling and achieve stable and reasonable core-genomes for similarity thresholds between 35 and 48. We receive similar results for the exclusive repertoire of the corynebacteria and mycobacteria, respectively. In conclusion, our results suggest that we may utilize the intrinsic information contained in the cluster size distribution, at least in the phylum actinobacteria, to deduce a reasonable density parameter for robust and accurate protein homology clustering. For future work with bacterial genomes, we suggest using BLAST E-values between 10^{-35} (optimistic) and 10^{-48} (conservative) when utilizing bi-directional BLAST hits only for homology detection. Remarkably, the same range is also suggested by comparing the agreement of our clustering result with the results from the OMA project.

Our method, however, is limited by the level of biological diversity amongst the set of species to be studied. As we only use the intrinsic information that is "hidden" in the dataset, we rely on a certain level of homogeneity. Hence, we can expect a reduced accuracy for more diverse sets of genomes.

Results of this Section

- Discovery of pathogenicity specific core-genomes.
- Construction of a whole genome repertoire phyologenetic tree of the actinobacterial CMNR group.
- Reliable and robust method for estimating a density parameter for protein homology detection.
- Our results suggest that 10^{-48} as optimal BLAST E-value cutoff for microbial protein homology detection.

8. Discussion

In the framework of this thesis, three main objectives were accomplished:

1. Extend TransClust for coping with missing values
2. Development of ClustEval for automatizing and unifying cluster analyses
3. The integration of Clustering in real-world biological examples

As the two biological studies are serving as examples for the application of clustering in systems biology and are mostly self-contained, we decided to discuss them in the corresponding section in order to enhance readability of the thesis (refer to Subsection 7.1.3 on page 133 and 7.2.3 on page 143).

Whereas the remaining two objectives will be discussed in the overall context of the thesis as they are highly interconnected in the following two sections.

8.1. Missing Values

In the first method section of this thesis, we introduced a memory efficient method for splitting a large similarity file into connected components. This approach allows for handling almost arbitrarily large similarity files on standard desktop computers with limited main memory. The new cost-matrix creator seamlessly integrates into existing clustering pipelines as it is compatible with the existing TransClust. The ability to decompose large similarity files on modest computing hardware resulted in a significantly increased runtime of the cost-matrix creator. That means the problem with handling large similarity files was not solved; rather, the bottleneck was shifted from memory consumption towards runtime requirements. This fact emphasizes the need to address the problem of increasingly larger similarity files with a different approach.

As a consequence, the central concept we elaborated upon in this work is reducing the size of the similarity file in the first place. We achieved this by the strategic use of missing values thus omitting the most similarities from being ever calculated. This does not only reduce the size of the similarity file but also enables the usage of computationally expensive similarity functions even for large datasets. In the framework of this thesis, we have developed two different approaches to integrate missing values into Transitivity Clustering:

	Affinity Propagation	CFinder	ClusterONE	CMC	Hierarchical Clustering	K-means	Markov Clustering	RNSC	RRW	TransClust	TransClustMV	ActiveTransClust
Scalability	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓
Robustness	✓	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓
Integration of existing knowledge	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
Arbitrary-shaped clusters	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓
Minimal user-specified input	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	(✓)
Interpretable results	✗	(✓)	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓
Reproducibility of results	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✗	✗*
Missing Values	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Active Clustering	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Table 8.1.: Overview of features defined in Subsection 2.4.11 on page 53 which are fulfilled by the clustering tools.
*due to the randomized nature of the active clustering approach, see text for details.

1. By randomly omitting similarity calculations

2. By actively calculating only the most important similarities

The first method called "TransClust with Randomly Missing Values" (TransClustMV) extends TransClust and only requires a different method for the creation of the cost-matrices. TransClustMV can handle single missing values as well as entirely missing one-vs.-all queries, i.e., the absence of entire stripes in the similarity matrix. The latter approach is more effective in terms of memory usage, as it does not require the explicit storage of the missing values within the similarity file but only requires a list of the omitted one-vs.-all queries. On our websites, we provide an implementation for the direct processing of FASTA files (using BLAST as similarity function); nevertheless, it can be implemented for different similarity functions as well.

The downside of this approach is that it only randomly choses the missing values (except the researcher includes prior knowledge by providing a list of objects used for the one-vs.-all queries) and thus we can hardly guarantee any limits for the quality of the clustering result. Our tests on protein homology detection suggest that the results merely differentiate, even when using only a small fraction of the available similarities, but this might not necessarily be true for all problem instances. Nevertheless, we removed the tick from "Reproducible results" from the Summary Table 8.1 because the reported results of TransClustMV depend on the selection of the one-vs.-all queries.

Our second approach, ActiveTransClust, takes the idea of using missing values in the similarity file further. ActiveTransClust starts with a small fraction of known similarity values. Based on the clustering of these initial similarities, ActiveTransClust calculates only those missing values which are most promising for improving the clustering result in addition to the already available similarities. Because of the additional process of calculating selected missing similarities, the user has to set more parameters compared to standard TransClust. We have set these parameters to well-evaluated default values which should produce good results while maintaining fast runtimes. Nevertheless, in Table 8.1, which summarizes all discussed clustering algorithms, we place the tick for "Minimum user-specified input" within in parentheses because some problems might require a deeper fine-tuning of the parameters. Furthermore, as ActiveTransClust also requires a randomized initialization step, we removed the tick from "Reproducible results" as we cannot guarantee that two runs report the same result. Nevertheless, our tests have shown that the variance between the different runs is very small.

To summarize (refer to Table 8.1), TransClustMV does fulfill all requirements of a state of the art clustering tool except the active clustering and the reproducibility of the results, whereas ActiveTransClust performs the active clustering by means of a more complex procedure. Furthermore, both methods are also dependent on acceptance within the scientific community. Results based on only a fraction of the similarities need to be acknowledged by the researchers as a valid clustering result. Otherwise, these tools, regardless of how accurate they are, will not be considered as real alternatives to clustering approaches using all similarities. Therefore, more studies with even larger datasets and computationally expensive similarity functions must be conducted in order to convince the scientific community. Some suggestions for such studies are outlined in the Outlook Chapter on page 159.

In the framework of this thesis, we have only used datasets which would not necessarily require an active clustering approach as they are still small enough for a classical approach. Additionally, we restricted ourselves to the usage of BLAST E-values as similarity function. We deliberately chose this approach as right now, demonstrating the feasibility of the active clustering approach achieving high quality results is the most important task. That of course means that we require comparable results from "classical" clustering approaches on well-known datasets using an established similarity measure. The results show that ActiveTransClust can produce high quality results with utilizing only a fraction of the available similarity information. Based on these results, we now can go large-scale and put reasonable confidence in reported future results.

Nevertheless, the influence of the active approach seems not that significant when looking at the F-measure only, especially on the non-artificial datasets. As already discussed in the chapter about ActiveTransClust (Chapter 5 on page 85), this is mainly due to two reasons: (1) large clusters, which are identified reliably, dominate the F-measure and (2) the tremendous amount of indirectly missing values in the similarity files (those values set to a cut-off value λ). We are convinced that the active approach is in fact improving the clustering quality, especially regarding small clus-

	ELKI	jClust	KNIME	RapidMiner	WEKA	ClustEval
Integration of Existing Tools	(✔)	(✔)	(✔)	(✔)	(✔)	✔
Data analysis	(✔)	✗	✔	✔	✔	✔
Cluster Evaluation	✗	✗	✗	✗	✗	✔
Parameter Optimization	✗	✗	(✔)	(✔)	✗	✔
Extensibility	✔	✗	✔	✔	✔	✔
Runtime Efficiency	✗	✗	(✔)	✗	✔	✔
Availability	(✔)	(✔)	(✔)	(✔)	(✔)	✔

Table 8.2.: Overview of features defined in Subsection 2.5.6 on page 60 which are fulfilled by the different evaluation frameworks.

ters, compared to the TransClustMV approach even though this fact is only poorly represented by the F-measure. When using the artificial datasets, we observe a much better performance as the cluster size distribution ensures that the large clusters do not contain the majority of the objects of the dataset. Furthermore, the artificial datasets also use a similarity measure with the same detection sensitivity for very dissimilar objects (i.e., not producing indirectly missing values), which increases the effectiveness of the algorithm. Generally, we followed a heuristic approach for integrating missing values into TransClust. We can not give conditions or properties of the dataset which assure that ActiveTransClust or TransClustMV report the optimal result or inverted, we can not give suggestions when the usage leads to poor results. To summarize, ActiveTransClust does improve the results over TransClustMV (at least for protein homology prediction) but works best when the similarity function is precise over the whole range of possible values (e.g., does not simply cut-off at a certain point of dissimilarity).

8.2. ClustEval

ClustEval is the first integrated evaluation platform specifically designed for clustering analyses. Table 8.2 summarizes the features of ClustEval in comparison to other evaluation frameworks. With ClustEval, we attempt to assist researchers in conducting a cluster analysis. We suggest a standard input and output format which is easily generated, parsed, and sufficiently feature-rich to address most use cases arising while clustering. At the moment, this file format is only supported by ActiveTransClust; for all other tools, parsers are required. ClustEval already provides parsers for the most commonly used tools in bioinformatics.

ClustEval also performs standardized and automated parameter training in order to enable transparent, reproducible, and fair studies. Nevertheless, no automated

threshold probing can replace the experience of experts. Thus, even though the threshold probing is fair in terms of the strategy applied to find the threshold, not every strategy necessarily fits to every clustering tool to the same extent. One can argue that even with ClustEval, there can be a bias introduced towards tools harmonizing with the chosen threshold probing best. Furthermore, such automated probing uses "only" a statistic measure for judging the quality of the tested parameter set whereas an expert might rule out an entire range of parameter sets by looking at the reported results.

ClustEval was designed to be as flexible as possible. That enables the integration of many different tools and allows the framework to be extended for future tools as well. On the other hand, exactly this flexibility necessitates configuration work of the researcher. Although we designed ClustEval to support practitioners in conducting a clustering study, they have to weigh the initialization costs (in terms of becoming familiar with the framework, installation, and configuration time) over the anticipated time saved during the actual study. Thus, for some studies, it is more efficient to conduct the study manually rather than using ClustEval. We attempted to reduce the set-up costs as much as possible, e.g., by providing an image of a Linux virtual machine with a fully functional pre-configured ClustEval framework; nevertheless, the complexity of the framework cannot be ignored, especially when the web front end for the result presentation is required.

Furthermore, ClustEval requires Linux and thereby rules out all clustering tools only available for a different operating system. Even though most tools are available for Linux, some practitioners use different operating systems. As ClustEval executes command line programs, it restricts the usage of tools being provided exclusively with a GUI or integrated into other frameworks, e.g., in Cytoscape (as an example, MCODE[10] is only available as Cytoscape plug-in[1]). We decided to evoke binaries because of the following advantages: (1) We can use the reference implementation of the tool, (2) including an updated version of a tool is only a matter of replacing a binary, (3) we are independent from the original programming language as long as there is a Linux version of the tool, and (4) we can also use closed-source tools without limitation. In our view, these advantages outweigh the advantages of directly including the different algorithms into ClustEval.

ClustEval provides a good overview of existing tools and their performance on different datasets which was especially tailored to the non-expert who wants to conduct a clustering study. Here, ClustEval should be regarded as a guide but it cannot reflect all possible options available for conducting an analysis. That can lead to a false confidence in terms of what is the best tool and the best parameter setting for a certain task if the results presented on ClustEval are not used with precaution.

[1] Of course, if these tools are open-source, they can be integrated in a modified version. ClustEval is not bound to execute command line tools.

9. Conclusion

In the course of this thesis, we have demonstrated the power and limits of cluster analyses in bioinformatics. We have elaborated on two main challenges which in our opinion are the currently most severe restrictions:

1. We have broadened the more and more apparent bottleneck of the similarity calculation inflicted by the increasing dataset size by exploiting missing values by introducing TransClustMV and ActiveTransClust.

2. We have eased the utilization of the complex and confusing field of clustering with its plethora of algorithms and methods by introducing ClustEval.

Regarding the first point, we have identified the similarity files and especially the calculation of all pairwise similarities as a bottleneck. As we have laid out in the introduction, this bottleneck will become even more severe with the growing amount of available biological datasets. As of yet, the application of complex similarity functions in terms of computational time (e.g., protein structure prediction and comparison) limits the size of feasible dataset drastically. We have tackled this challenge by introducing missing values to the well-established clustering tool TransClust. This approach does not need all pairwise similarities; for protein sequence clustering, we were able to show that in fact only a fraction of all similarities are required. Furthermore, we extended this approach by introducing a feedback loop when we combine the clustering itself and the calculation of the similarities. Here, we base a clustering on few similarities and calculate specifically those similarities which promise the best effect on the clustering result. With that, we were able to reduce the required amount of information even further while still delivering the same result quality. To conclude, with the approaches presented in this thesis, the bottleneck of the similarity calculation was widened and enables large-scale studies with complex similarity functions in future applications. The user is given the choice: If the calculation of an entire all-vs.-all similarity file is no longer suitable, he can either choose TransClustMV which is faster but requires more similarities for the same quality, or ActiveTransClust if the calculations of the similarities are sufficiently time consuming to outweigh the increased runtime of ActiveTransClust compared to TransClustMV.

Application of the new Clustering Approaches

Classical TransClust Classical TransClust can be applied whenever the similarity file is already calculated.

TransClustMV Best used when the similarity function is computationally simple but the amount of objects render a complete calculation impossible.

ActiveTransClust In cases of very complex and time-consuming similarity functions such that the time saved in similarity file creation outweighs the runtime advantage of TransClustMV.

In the second main part of the thesis, we developed ClustEval. As clustering is a long standing problem in computer science, there exist many different approaches and tools to calculate a clustering. The amount of possibilities is quite overwhelming especially for non-experts. Furthermore, the correct usage of a clustering tool requires extensive knowledge in the underlying algorithm to be efficient and produce the best possible result. ClustEval addresses many of these problems. First, the non-expert can inform himself through a useful overview of the existing tools, their performance on different datasets, and the best parameter setting for certain applications. The expert is supported by ClustEval by means of an automatization of the laborious parts of a cluster analysis, like the parameter training of several algorithms. Here, we proposed standards for input and output files as well as for parameter training and result evaluation. Results based on an evaluation standard render the results more significant and easier to compare. The platform is open for the community to contribute and has the potential to become the "one-stop-shop" for conducting cluster-analysis.

The Clustering Evaluation Platform ClustEval

- One-stop-shop for practitioners conducting cluster analyses.
- Automatization and unification of the entire clustering process including bias-free parameter training.
- Easily extendible for future developments.

10. Outlook

To finish the work, we want to outline some possible future developments in order to further broaden the capabilities of clustering algorithms in bioinformatics.

Multiple Disks
The presented method for creating cost-matrices by means of utilizing background storage right now assumes a simple one disk environment. An increasing number of modest computing hardware is equipped with more than one hard disk, thus the efficient usage of multiple disk systems will be of increasing importance in the future. Furthermore, as the main limiting factor of the cost-matrix creation process is the I/O load, an compressed format for cost-matrices could further reduce the I/O load and thus lead to a drop in the runtime.

External Cost-Matrices
The clustering algorithm of TransClust still relies on the entire cost-matrix being stored in the main memory. Even though the creation of large cost-matrices poses no problem anymore, the actual clustering of such a matrix does. Here, methods for utilizing background storage during the clustering process may ease the problem. During the clustering, the costs for the edge insertions/removals are not queried in a predefined order, but show almost random-access behavior. Therefore, efficient methods for random access to data on the background storage need to be implemented. Possible solutions are the usage of databases or external hash tables, e.g., as described in [126]. Implementing these steps would allow for almost arbitrarily large clustering projects using TransClust.

Complex Similarity Functions
We have demonstrated the capabilities and the quality of TransClustMV and ActiveTransClust in this thesis using well-known datasets. This foundation allows for the the development of projects specifically benefiting from these new capabilities and going beyond the proof-of-concept phase as done in this work. Especially the usage of complex similarity functions promises to generate new insights. As an example, entire regulatory networks could be used as objects combined with their degree of alignment (which is in fact a NP-hard problem in itself) as a similarity function in order to discover closely related sets of regulatory networks of different species. Another example is the usage of protein homology detection using 3D protein structure

alignment, e.g. 3D-BLAST[1] [78]. To summarize, the computational complexity of the similarity function does not influence the overall runtime of a cluster analysis that drastically anymore.

Hybrid Similarity Functions
With the rise of such complex similarity functions, the development of a "hybrid approach" could further speed-up the process while increasing the result quality. In a hybrid approach, we combine a fast similarity function with a computationally expensive but more accurate one. Thus, the active clustering approach uses the information from the fast similarity function in order to identify critical clusters. For those critical clusters, the expensive similarity function is applied for calculating more reliable similarities.

Online Clustering
A further extension to the active clustering approach could be the introduction of an online mode. That means that not all objects of the dataset are known right away, but are added at a later point in time without discarding the already performed cluster analysis. This approach is compatible with the methods already developed in this chapter, as the newly added objects will simply be treated as singletons with no information so far and thus are automatically ranked high for possible improvements. When using the landmark method, the newly added objects can be integrated not with an one-vs.-all query but a still efficient "one-vs.-new" query using the existing landmarks.

Practical Implications
To outline the new possibilities with the clustering approaches presented here, a system comparable to EHECRegNet could now be realized hosting more species. The usage of several model organisms (instead of just one) together with target organisms spread all over the different phyla of bacteria could create new levels of evidence for conserved regulations. With the active approach, such a project could be realized using an intermediate clustering and then improving the quality of the database over time, increasing the confidence in the predicted regulations. Together with the online approach, results for newly added organisms could be already utilized after a couple of minutes.

To conclude, a high quality similarity function is one of the most important parts of a cluster analysis. A clustering based on an infeasible similarity function can never produce meaningful results, regardless of the used clustering algorithm. With the work presented here, the number of applicable similarity functions is increased (as the runtime of a similarity calculation is less important than before) and also allows for the development of novel similarity approaches. In the meantime, several existing systems can be supported by a larger data basis, especially when implementing the suggested improvements.

ClustEval Beyond Bioinformatics
ClustEval, the integrated clustering framework developed in this thesis can also be

[1] A single protein pair query can take up to 4 seconds, depending on the search depth used.

extended to broaden the range of possible application areas. The scope of the framework can also include clustering methods popular in other disciplines besides bioinformatics. The same applies for the accompanying datasets. This will enable the researchers to have an unbiased and realistic view on the performance of the most important clustering algorithms and will assist them in the selection process of an appropriate tool for their dataset.

Automated Tool Suggestion
An increasing number of datasets integrated into ClustEval, especially those with a reliable gold standard, will allow for new types analyses. As for every dataset many properties are known, we can relate the performance of the different clustering tools to these properties. This can be realized with a machine learning approach using the dataset properties as features. With that knowledge, we will be able to predict the anticipated performance of each clustering tool for an unknown dataset once its properties are calculated. That would further ease the difficult process of choosing the best possible clustering tool as ClustEval will provide an objective suggestion.

ClustEval as an Online Clustering Service
A further step to improve the user-friendliness of ClustEval would be the transformation of the framework to an online service. That means, a user can upload a dataset and will automatically receive his clustering result retrieved from the desired clustering tools. All that would not require any installation or deep knowledge concerning the different tools by the user. Such a step will be especially important for non-computer experts as the necessity of running an unknown tool locally vanishes.

ClustEval Outside of Clustering
The modular and flexible design of ClustEval also allows for adaption to different tasks than clustering. The internal structure of ClustEval can be used wherever different tools and measures are supposed to run on numerous datasets. For example, we could provide a ClustEval fork specialized for classification or graph editing which only requires the implementation of additional parsers and measures for assessing the result quality.

All these implications will further improve the acceptance and importance of ClustEval outside of the clustering community.

Bibliography

[1] E. Achtert, C. Böhm, and P. Kröger. DeLi-Clu: boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In *Advances in Knowledge Discovery and Data Mining*, pages 119–128. Springer, 2006.

[2] E. Achtert, H.-P. Kriegel, and A. Zimek. ELKI: a software system for evaluation of subspace clustering algorithms. In *Scientific and Statistical Database Management*, pages 580–585. Springer, 2008.

[3] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, Apr 2006.

[4] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, Jun 1999.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, Oct 1990.

[6] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, Sep 1997.

[7] B. Andreopoulos, A. An, X. Wang, and M. Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314, May 2009.

[8] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. *ACM SIGMOD Record*, 28(2):49–60, 1999.

[9] G. J. Babu and E. D. Feigelson. Statistical Challenges in Modern Astronomy II. In *Statistical Challenges in Modern Astronomy II*, volume 1, 1997.

[10] G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, Jan 2003.

[11] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1068–1077. Society for Industrial and Applied Mathematics, 2009.

[12] T. Barrett, S. E. Wilhite, P. Ledoux, C. Evangelista, I. F. Kim, M. Tomashevsky, K. A. Marshall, K. H. Phillippy, P. M. Sherman, M. Holko, A. Yefanov, H. Lee, N. Zhang, C. L. Robertson, N. Serova, S. Davis, and A. Soboleva. NCBI GEO: archive for functional genomics data sets–update. *Nucleic Acids Research*, 41(Database issue):D991–D995, Jan 2013.

[13] J. Baumbach. On the power and limits of evolutionary conservation–unraveling bacterial gene regulatory networks. *Nucleic Acids Research*, 38(22):7877–7884, Dec 2010.

[14] J. Baumbach, S. Rahmann, and A. Tauch. Reliable transfer of transcriptional gene regulatory networks between taxonomically related organisms. *BMC Systems Biology*, 3:8, 2009.

[15] J. Baumbach, T. Wittkop, C. K. Kleindt, and A. Tauch. Integrated analysis and reconstruction of microbial transcriptional gene regulatory networks using CoryneRegNet. *Nature Protocols*, 4(6):992–1005, 2009.

[16] M. Beckstette, R. Homann, R. Giegerich, and S. Kurtz. Fast index based algorithms and software for matching position specific scoring matrices. *BMC Bioinformatics*, 7:389, 2006.

[17] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999.

[18] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. GenBank. *Nucleic Acids Research*, 41(Database issue):D36–D42, Jan 2013.

[19] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 36(Database issue):D25–D30, Jan 2008.

[20] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. *KNIME: The Konstanz information miner*. Springer, 2008.

[21] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 28(3):301–315, 1998.

[22] S. Brohée and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, 2006.

[23] S. D. Brown, J. A. Gerlt, J. L. Seffernick, and P. C. Babbitt. A gold standard set of mechanistically diverse enzyme superfamilies. *Genome Biology*, 7(1):R8, 2006.

[24] E. Brzuszkiewicz, A. Thürmer, J. Schuldes, A. Leimbach, H. Liesegang, F.-D. Meyer, J. Boelter, H. Petersen, G. Gottschalk, and R. Daniel. Genome sequence analyses of two isolates from the recent Escherichia coli outbreak in Germany reveal the emergence of a new pathotype: Entero-Aggregative-Haemorrhagic Escherichia coli (EAHEC). *Archives of Microbiology*, 193(12):883–891, Dec 2011.

[25] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden. BLAST+: architecture and applications. *BMC Bioinformatics*, 10:421, 2009.

[26] G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992.

[27] C. Chakrapani. *Statistics in market research*. Hodder Arnold, 2004.

[28] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[29] Y. Chen, K. D. Reilly, A. P. Sprague, and Z. Guan. SEQOPTICS: a protein sequence clustering system. *BMC Bioinformatics*, 7 Suppl 4:S10, 2006.

[30] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[31] A. Clauset, C. R. Shalizi, and M. E. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[32] D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1(2):224–227, 1979.

[33] P. S. Dehal, M. P. Joachimiak, M. N. Price, J. T. Bates, J. K. Baumohl, D. Chivian, G. D. Friedland, K. H. Huang, K. Keller, P. S. Novichkov, I. L. Dubchak, E. J. Alm, and A. P. Arkin. MicrobesOnline: an integrated portal for comparative and functional genomics. *Nucleic Acids Research*, 38(Database issue):D396–D400, Jan 2010.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[35] C. Dessimoz, B. Boeckmann, A. C. J. Roth, and G. H. Gonnet. Detecting non-orthology in the COGs database and other approaches grouping orthologs using genome-specific best hits. *Nucleic Acids Research*, 34(11):3309–3316, 2006.

[36] C. Dessimoz, G. Cannarozzi, M. Gil, D. Margadant, A. Roth, A. Schneider, and G. H. Gonnet. OMA, a comprehensive, automated project for the identification of orthologs from complete genome data: introduction and first achievements. In *Comparative Genomics*, pages 61–72. Springer, 2005.

[37] F. A. Dorella, L. G. C. Pacheco, S. C. Oliveira, A. Miyoshi, and V. Azevedo. Corynebacterium pseudotuberculosis: microbiology, biochemical properties, pathogenesis and molecular studies of virulence. *Veterinary Research*, 37(2):201–218, 2006.

[38] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[39] N. Eagle, M. Macy, and R. Claxton. Network diversity and economic development. *Science*, 328(5981):1029–1031, 2010.

[40] R. Edgar, M. Domrachev, and A. E. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, Jan 2002.

[41] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, Dec 1998.

[42] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, Apr 2002.

[43] P. Erdös and A. Renyi. On random graphs I. *Publicationes Mathematicae-Debrecen*, 6:290–297, 1959.

[44] L. Ertoz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the Third SIAM International Conference on Data Mining*, page 47, 2003.

[45] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[46] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD Explorations Newsletter*, 4(1):65–75, 2002.

[47] B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd, 5th edition edition, 2011.

[48] I. Frades and R. Matthiesen. Overview on techniques in cluster analysis. *Methods in Molecular Biology*, 593:81–107, 2010.

[49] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, Feb 2007.

[50] S. Gama-Castro, H. Salgado, M. Peralta-Gil, A. Santos-Zavaleta, L. Muñiz Rascado, H. Solano-Lira, V. Jimenez-Jacinto, V. Weiss, J. S. García-Sotelo, A. López-Fuentes, L. Porrón-Sotelo, S. Alquicira-Hernández, A. Medina-Rivera, I. Martínez-Flores, K. Alquicira-Hernández, R. Martínez-Adame, C. Bonavides-Martínez, J. Miranda-Ríos, A. M. Huerta, A. Mendoza-Vargas, L. Collado-Torres, B. Taboada, L. Vega-Alvarado, M. Olvera, L. Olvera, R. Grande, E. Morett, and J. Collado-Vides. RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units). *Nucleic Acids Research*, 39(Database issue):D98–105, Jan 2011.

[51] B. Gao and R. S. Gupta. Microbial systematics in the post-genomics era. *Antonie van Leeuwenhoek*, 101(1):45–54, Jan 2012.

[52] B. Gao and R. S. Gupta. Phylogenetic framework and molecular signatures for the main clades of the phylum Actinobacteria. *Microbiology and Molecular Biology Reviews*, 76(1):66–112, Mar 2012.

[53] B. Gao, R. Paramanathan, and R. S. Gupta. Signature proteins that are distinctive characteristics of Actinobacteria and their subgroups. *Antonie van Leeuwenhoek*, 90(1):69–91, Jul 2006.

[54] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, Oct 1999.

[55] A. D. Gordon. A review of hierarchical classification. *Journal of the Royal Statistical Society. Series A (General)*, pages 119–137, 1987.

[56] J. C. Gower and P. Legendre. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, 1986.

[57] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[58] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[59] J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, Aug 2005.

[60] T. Hastie, R. Tibshirani, and J. Friedman. Unsupervised Learning. In *The Elements of Statistical Learning*, Springer Series in Statistics, pages 485–585. Springer New York, 2009.

[61] M. Hauser, C. E. Mayer, and J. Söding. kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinformatics*, 14:248, 2013.

[62] S. D. Hooper and P. Bork. Medusa: a simple tool for interaction graph analysis. *Bioinformatics*, 21(24):4432–4433, Dec 2005.

[63] J. B. Kaper and M. A. Karmali. The continuing evolution of a bacterial pathogen. *Proceedings of the National Academy of Sciences of the United States of America*, 105(12):4535–4536, Mar 2008.

[64] K. A. Karberg, G. J. Olsen, and J. J. Davis. Similarity of genes horizontally acquired by Escherichia coli and Salmonella enterica is evidence of a supraspecies pangenome. *Proceedings of the National Academy of Sciences of the United States of America*, 108(50):20154–20159, Dec 2011.

[65] H. Karch. The role of virulence factors in enterohemorrhagic Escherichia coli (EHEC)–associated hemolytic-uremic syndrome. *Seminars in Thrombosis and Hemostasis*, 27(3):207–213, Jun 2001.

[66] A. D. King, N. Przulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, Nov 2004.

[67] A. Krause, J. Stoye, and M. Vingron. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, 6:15, 2005.

[68] J. Krawczyk, T. A. Kohl, A. Goesmann, J. Kalinowski, and J. Baumbach. From Corynebacterium glutamicum to Mycobacterium tuberculosis–towards transfers of gene regulatory networks and integrated data analyses with MycoRegNet. *Nucleic Acids Research*, 37(14):e97, Aug 2009.

[69] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[70] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. *Bioinformatics*, 24(5):719–720, 2008.

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

[72] G. Liu, L. Wong, and H. N. Chua. Complex discovery from weighted PPI networks. *Bioinformatics*, 25(15):1891–1897, Aug 2009.

[73] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu. Understanding and enhancement of internal clustering validation measures. *Cybernetics, IEEE Transactions on*, 43(3):982–994, Jun 2013.

[74] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[75] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, number 281-297, page 14. California, USA, 1967.

[76] K. Macropol, T. Can, and A. K. Singh. RRW: repeated random walks on genome-scale protein networks for local cluster discovery. *BMC Bioinformatics*, 10:283, 2009.

[77] L. M. Mallory-Greenough, J. D. Greenough, and J. V. Owen. New data for old pots: Trace-element characterization of ancient Egyptian pottery using ICP-MS. *Journal of Archaeological Science*, 25(1):85–97, 1998.

[78] L. Mavridis and D. W. Ritchie. 3D-blast: 3D protein structure alignment, comparison, and classification using spherical polar Fourier correlations. *Pacific Symposium on Biocomputing*, pages 281–292, 2010.

[79] J. L. Mellies, A. M. S. Barron, and A. M. Carmona. Enteropathogenic and enterohemorrhagic Escherichia coli virulence gene regulation. *Infection and Immunity*, 75(9):4199–4210, Sep 2007.

[80] M. L. Metzker. Sequencing technologies - the next generation. *Nature Reviews Genetics*, 11(1):31–46, Jan 2010.

[81] V. Miao and J. Davies. Actinobacteria: the good, the bad, and the ugly. *Antonie van Leeuwenhoek*, 98(2):143–150, Aug 2010.

[82] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940. ACM, 2006.

[83] G. W. Milligan. Clustering validation: results and implications for applied analyses. *Clustering and Classification*, 1:341–375, 1996.

[84] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, Apr 1995.

[85] J. L. Myers, A. D. Well, and R. F. Lorch. *Research design and statistical analysis*. Routledge, 2010.

[86] N.C.B.I. Resource Coordinators. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 41(Database issue):D8–D20, Jan 2013.

[87] T. Nepusz, H. Yu, and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471–472, May 2012.

[88] M. E. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118, 2001.

[89] R. Nugent and M. Meila. An overview of clustering applied to molecular biology. *Methods in Molecular Biology*, 620:369–404, 2010.

[90] M. G. Omran, A. P. Engelbrecht, and A. Salman. An overview of clustering methods. *Intelligent Data Analysis*, 11(6):583–605, 2007.

[91] A. Paccanaro, J. A. Casbon, and M. A. Saqi. Spectral clustering of protein sequences. *Nucleic Acids Research*, 34(5):1571–1580, 2006.

[92] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, Jun 2005.

[93] J. Pauling, R. Röttger, A. Neuner, H. Salgado, J. Collado-Vides, P. Kalaghatgi, V. Azevedo, A. Tauch, A. Pühler, and J. Baumbach. On the trail of EHEC/EAEC–unraveling the gene regulatory networks of human pathogenic Escherichia coli bacteria. *Integrative Biology*, 4(7):728–733, Jul 2012.

[94] J. Pauling, R. Röttger, A. Tauch, V. Azevedo, and J. Baumbach. CoryneRegNet 6.0–Updated database content, new analysis methods and novel features focusing on community demands. *Nucleic Acids Research*, 40(Database issue):D610–D614, Jan 2012.

[95] G. A. Pavlopoulos, C. N. Moschopoulos, S. D. Hooper, R. Schneider, and S. Kossida. jClust: a clustering and visualization toolbox. *Bioinformatics*, 25(15):1994–1996, Aug 2009.

[96] M. Pertea, K. Ayanbule, M. Smedinghoff, and S. L. Salzberg. OperonDB: a comprehensive database of predicted operons in microbial genomes. *Nucleic Acids Research*, 37(Database issue):D479–D482, Jan 2009.

[97] S. Powell, D. Szklarczyk, K. Trachana, A. Roth, M. Kuhn, J. Muller, R. Arnold, T. Rattei, I. Letunic, T. Doerks, L. J. Jensen, C. von Mering, and P. Bork. eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Research*, 40(Database issue):D284–D289, Jan 2012.

[98] S. Rahmann, T. Wittkop, J. Baumbach, M. Martin, A. Truss, and S. Böcker. Exact and heuristic algorithms for weighted cluster editing. *Computational Systems Bioinformatics Conference*, 6:391–401, 2007.

[99] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of The American Statistical Association*, 66(336):846–850, 1971.

[100] R. Röttger, P. Kalaghatgi, P. Sun, S. d. C. Soares, V. Azevedo, T. Wittkop, and J. Baumbach. Density parameter estimation for finding clusters of homologous proteins–tracing actinobacterial pathogenicity lifestyles. *Bioinformatics*, 29(2):215–222, Jan 2013.

[101] R. Röttger, C. Kreutzer, T. D. Vu, T. Wittkop, and J. Baumbach. Online Transitivity Clustering of Biological Data with Missing Values. In *GCB*, pages 57–68, 2012.

[102] R. Röttger, U. Rückert, J. Taubert, and J. Baumbach. How Little Do We Actually Know? On the Size of Gene Regulatory Networks. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 9(5):1293–1300, 2012.

[103] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[104] G. Salton. Developments in automatic text retrieval. *Science*, 253(5023):974–980, Aug 1991.

[105] J. Shendure and H. Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, Oct 2008.

[106] R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[107] E. Stackebrandt. Phylogeny Based on 16SrRNA/DNA. *eLS*, 2009.

[108] E. Stackebrandt, R. Murray, and H. Trüper. Proteobacteria classis nov., a name for the phylogenetic taxon that includes the "purple bacteria and their relatives". *International Journal of Systematic Bacteriology*, 38(3):321–325, 1988.

[109] A. Strehl. *Relationship-based clustering and cluster ensembles for high-dimensional data mining.* PhD thesis, 2002.

[110] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.

[111] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[112] S. M. van Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, 2000.

[113] M. Ventura, C. Canchaya, A. Tauch, G. Chandra, G. F. Fitzgerald, K. F. Chater, and D. van Sinderen. Genomics of Actinobacteria: tracing the evolutionary history of an ancient phylum. *Microbiology and Molecular Biology Reviews*, 71(3):495–548, Sep 2007.

[114] A. I. Verkamo. Performance comparison of distributive and mergesort as external sorting algorithms. *Journal of Systems and Software*, 10(3):187–200, 1989.

[115] J. S. Vitter. Algorithms and data structures for external memory. *Foundations and Trends® in Theoretical Computer Science*, 2(4):305–474, 2008.

[116] K. Voevodski, M.-F. Balcan, H. Roglin, S.-H. Teng, and Y. Xia. Efficient clustering with limited distance information. *arXiv preprint arXiv:1009.5168*, 2010.

[117] D. R. White and F. Harary. The cohesiveness of blocks in social networks: Node connectivity and conditional density. *Sociological Methodology*, 31(1):305–359, 2001.

[118] L. H. Williamson. Caseous lymphadenitis in small ruminants. *Veterinary Clinics of North America. Food Animal Practice*, 17(2):359–71, vii, Jul 2001.

[119] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[120] T. Wittkop. *Clustering biological data by unraveling hidden transitive substructures*. PhD thesis, Center for Biotechnology, Universität Bielefeld, 2010.

[121] T. Wittkop, J. Baumbach, F. P. Lobo, and S. Rahmann. Large scale clustering of protein sequences with FORCE -A layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, 8:396, 2007.

[122] T. Wittkop, D. Emig, S. Lange, S. Rahmann, M. Albrecht, J. H. Morris, S. Böcker, J. Stoye, and J. Baumbach. Partitioning biological data with transitivity clustering. *Nature Methods*, 7(6):419–420, Jun 2010.

[123] T. Wittkop, D. Emig, A. Truss, M. Albrecht, S. Böcker, and J. Baumbach. Comprehensive cluster analysis with Transitivity Clustering. *Nature Protocols*, 6(3):285–295, Mar 2011.

[124] T. Wittkop, S. Rahmann, and J. Baumbach. Efficient online transcription factor binding site adjustment by integrating transitive graph projection with MoRAine 2.0. *Journal of Integrative Bioinformatics*, 7(3), 2010.

[125] T. Wittkop, S. Rahmann, R. Röttger, S. Böcker, and J. Baumbach. Extension and robustness of transitivity clustering for protein–protein interaction network analysis. *Internet Mathematics*, 7(4):255–273, 2011.

[126] P. Woelfel. Maintaining external memory efficient hash tables. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 508–519. Springer, 2006.

[127] R. Xu and D. C. Wunsch. Clustering Algorithms in Biomedical Research: A Review. *Biomedical Engineering, IEEE Reviews in*, 3:120–154, 2010.

Nomenclature

ARFF Attribute Relation File Format, page 59

BeH Best Hit, page 30

BFS Breadth-First Search, page 66

BLAST Basic Local Alignment Search Tool, page 29

CFinder Clique Finder, page 42

ClusterONE Clustering with Overlapping Neighborhood Expansion, page 43

CMC Clustering based on Maximal Cliques, page 43

CMC Cost-Matrix Creator, page 67

Cov Coverage, page 30

DFS Depth-First Search, page 66

EHEC Enterohemorrhagic E. coli, page 128

Elki Environment for Developing KDD-Applications Supported by Index-Structures, page 58

FPT Fixed Parameter Tractable, page 52

GEO GenExpressionOmnibus, page 14

HSP High Scoring Pair, page 30

HUS hemolytic uremic syndrome, page 128

IDE Integrated Development Environment, page 58

KNIME The Konstanz Information Miner, page 58

MCL Markov Clustering, page 47

RNSC Restricted Neighborhood Search Clustering, page 47

RRW Repeated Random Walks, page 48

SoH Sum of Hits, page 30

WEKA Waikato Environment for Knowledge Analysis, page 59

WTGPP Weighted Transitive Graph Projection Problem / cluster editing problem, page 49

YALE Yet Another Learning Environment, page 59

List of Figures

List of Tables

A. ClustEval

This appendix supplements the chapter about ClustEval. Here, we will give an overview of the files system structure and give examples for configuration files. For all options and details about ClustEval, refer to the technical documentation which can be found on the website clusteval.mpi-inf.mpg.de.

A.1. File System Structure of the Repository

The repository is the main data source for the ClustEval framework. The file structure is automatically generated by the server whenever a new repository is generated. The structure is as follows:

```
repository/
├── repository.config
├── data/
│   ├── configs/
│   │   ├── brown.dataconfig
│   │   └── ...
│   ├── datasets/
│   │   ├── configs/
│   │   ├── brown/
│   │   └── .../
│   └── goldstandards/
│       ├── configs/
│       ├── brown/
│       └── .../
├── programs/
│   ├── configs/
│   │   ├── transclust.config
│   │   └── ...
│   ├── TransClust/
│   │   └── TransClust.jar
│   └── .../
└── ...
```

```
...
├── results/
│   ├── <date>_<runconfig>/
│   │   ├── analyses
│   │   ├── clusters
│   │   ├── configs
│   │   ├── goldstandards
│   │   ├── inputs
│   │   └── logs
│   └── .../
├── runs/
│   ├── CaseStude_I.run
│   ├── CaseStude_II.run
│   ├── CaseStude_III.run
│   └── ...
└── supp/
    ├── clustering/
    ├── distanceMeasures/
    ├── formats/
    ├── generators/
    ├── statistics/
    └── .../
```

As seen in the listing, the repository consists mainly of five directories:

data This directory stores all datasets and gold standards. Each dataset is located in its own directory. In order to propagate the existence of a dataset, a configuration files has to be placed in the "`config`" subdirectory. The configuration files in the subdirectory "`data`" link the dataset and the gold standard together. The configuration files in the "`config`" subdirectories of "`dataset`" and "`goldstandard`" finally point to the actual data file. An example can be found in the Appendix A.2 on the next page.

programs Here, all binaries of the clustering tools are stored. Each program is put in its own directory and a configuration file has to be stored in the "`config`" subdirectory. An example for a program configuration can be found in the Appendix A.2 on the facing page.

results Every run creates a own subdirectory beginning with the execution date and the name of the executed run configuration. In each of these result directories, the user not only finds the results but also all used input files. This ensures reproducibility, even when the datasets in the "`data`" directory are changed, as everything required for a re-run is stored together with the results.

runs This directory only consists of run configuration files. An example can be found in the Appendix A.2 on the next page.

supp This directory contains all plug-ins for ClustEval. In every subdirectory "JAR" files are stored implementing a given Interface of ClustEval, e.g., the implementation of the Euclidean distance can be found at "`/repository/supp/distanceMeasures/EuclidianDistanceMeasure.jar`"

All directories are automatically scanned by the server for changes. These changes are then included into the Server and can be used right after copying the files into the proper directories.

A.2. Configuration File Examples

A.2.1. Program Configuration

The following listing displays the configuration for TransClust and its parameters.

```
1 program = TransClust/TransClust.jar
2 parameters = minT,maxT,T,tss,mode
3 optimizationParameters = T
4 compatibleDataSetFormats = RowSimDataSetFormat
5 outputFormat = TransClustRunResultFormat
6 alias = Transitivity Clustering
7
8 [invocationFormat]
9 invocationFormat = java -jar %e% -i %i% -sim %i% -gs %gs% -o %o% -minT
      %minT% -maxT %maxT% -tss %tss% -mode %mode% -verbose
10
11 [T]
12 desc = Threshold
13 type = 2
14 def = $(meanSimilarity)
15 minValue = $(minSimilarity)
16 maxValue = $(maxSimilarity)
17
18 [minT]
19 desc = min. Threshold
20 type = 2
21 def = 0.8
22
23 [maxT]
24 desc = max. Threshold
25 type = 2
26 def = 1.0
27
28 [tss]
29 desc = Threshold stepsize
30 type = 2
31 def = 0.01
32
33 [mode]
34 type = 1
35 def = 2
```

A.2.2. Dataset Configuration

A dataset has an entire hierarchy of configurations. In the "`data`" directory, the dataset is linked to its gold standard, if existent. An example for the Brown *et al.* dataset looks as follows:

```
datasetConfig = brown
goldstandardConfig = brown
```

The configuration file of the Brown *et al.* dataset in the "`dataset`" directory reads as follows:

```
datasetName = brown
datasetFile = brown_et_al.txt
```

Here, the link to the actual data file is established, which looks as follows:

```
// dataSetFormat = RowSimDataSetFormat
// dataSetType = ProteinSequenceSimilarityDataSetType
// dataSetFormatVersion = 1
// alias = brown
gi15801179      gi16129025      323.3062153431158
gi15801179      gi1346930       3.0969100130080562
gi15801179      gi3122654       3.0
...
```

The same scheme is applied for the gold standard file as well.

A.2.3. Run Configuration

The following listing displays the a sample configuration file used for case study I (see Subsection 6.4.1 on page 120):

```
programConfig = AffinityPropagation,Hierarchical_Clustering,K-means,MCL
    ,TransClust
dataConfig = synthetic_cuboid250,synthetic_cassini250,
    synthetic_spirals250
qualityMeasures = DunnIndexRClusteringQualityMeasure,
    DaviesBouldinIndexRClusteringQualityMeasure,
    SilhouetteValueRClusteringQualityMeasure,
    FmeasureClusteringQualityMeasure,
    JaccardIndexRClusteringQualityMeasure,
    RandIndexRClusteringQualityMeasure,
    SensitivityClusteringQualityMeasure
mode = parameter_optimization
optimizationMethod = LayeredDivisiveParameterOptimizationMethod
optimizationCriterion = FmeasureClusteringQualityMeasure
optimizationIterations = 1001

[TransClust]
optimizationParameters = T

[MCL]
optimizationParameters = I
```

```
14
15 [AffinityPropagation]
16 optimizationParameters = preference,dampfact,maxits,convits
17
18 [Hierarchical_Clustering]
19 optimizationParameters = k
20
21 [K-means]
22 optimizationParameters = k
```

In this file, all used datasets and clustering tools are specified. The "mode" specifies that ClustEval is supposed to perform a parameter optimization, using the "Layered Divisive Method" testing 1001 . The parameters are optimized to achieve the best F-measure. In the lower part of the configuration file, further details for the clustering algorithms are given. This listing only serves as example, for all options and details about ClustEval, refer to the technical documentation which can be found on the website of ClustEval clusteval.mpi-inf.mpg.de.

A.3. Integrated Dataset Measures

Besides the mentioned measures for the clustering quality, ClustEval provides the following properties for datasets:

Intra-vs.-Inter This property plots a digram showing the similarity distribution for similarities within clusters and between clusters. This measure is only available for datasets with a gold standard.

Overlap of Intra-vs.Inter This measure is only available for datasets with a gold standard. The measure calculates the proportion of the area of the overlap within the Intra-vs.-Inter plot.

Similarity Distribution Plots the distribution of the similarities.

Graph Adhesion The graph adhesion is a measure introduced by White and Harary in [117]. This measure describes the minimal sum of edge weights which need to be removed from the graph such that the graph is no longer strongly connected.

Graph Cohesion The analogous property to the adhesion, but this time describes the minimal number of nodes to be removed from the graph such that the graph is no longer strongly connected.

Graph Min-Cut The minimal sum of edge weights of edges required to be removed in order to decompose the graph in at least two connected components.

Clustering Coefficient The cluster coefficient is the ratio of triangles (i.e., three nodes with three edges, thus a clique) and "connected triplets" (three node connected by at least two edges). The coefficient C is calculated as

$$C = \frac{3 \cdot \#\text{triangles}}{\#\text{triplets}}.$$

The factor 3 is necessary, as every triangle is also counted as three triplets.

Node Degree Distribution Plots the node degree distribution. In case of weighted networks, the sum of the weights of all adjacent edges of a node are reported as its degree.

Graph Density The ratio of existing edges and the maximal possible number of edges (in a network with n Node, this would be $\frac{n(n-1)}{2}$).

Graph Diversity Measures the diversity of the node degrees (or the weight sum of all adjacent edges) utilizing the Shannon entropy. The approach follows the publication of Eagle *et al.* [39]. The normalized graph diversity is calculated as:

$$\begin{aligned} D &= \frac{1}{n}\sum_{u \in V} D(u) = \frac{1}{n}\sum \frac{H(u)}{\log(\deg(u))} \quad \text{with} \\ H(u) &= -\sum_{v \in V} w(u,v)\log(w(u,v)) \end{aligned}$$

with $\deg(u)$ being the degree of the node u and $H(u)$ the Shannon entropy. The term $w(u,v)$ is the weight of the edge uv.

Matrix Rank Calculates the rank of the similarity matrix, i.e., the number of linear independent rows of the matrix.

A.4. Case Study I - Additional Results

A.4.1. Dataset Properties

Figure A.1 displays the the Intra-vs.-Inter similarity distribution and the node degree distribution of the Cassini and the Spiral dataset. Table A.1 shows the remaining properties of both datasets. Noteworthy is the very high overlap of the Intra-vs.-Inter cluster similarity of the spiral dataset which is a prominent property of that dataset.

A.4.2. Cluster Results

Figure A.2 depicts the clustering result for the remaining two datasets of case study I.

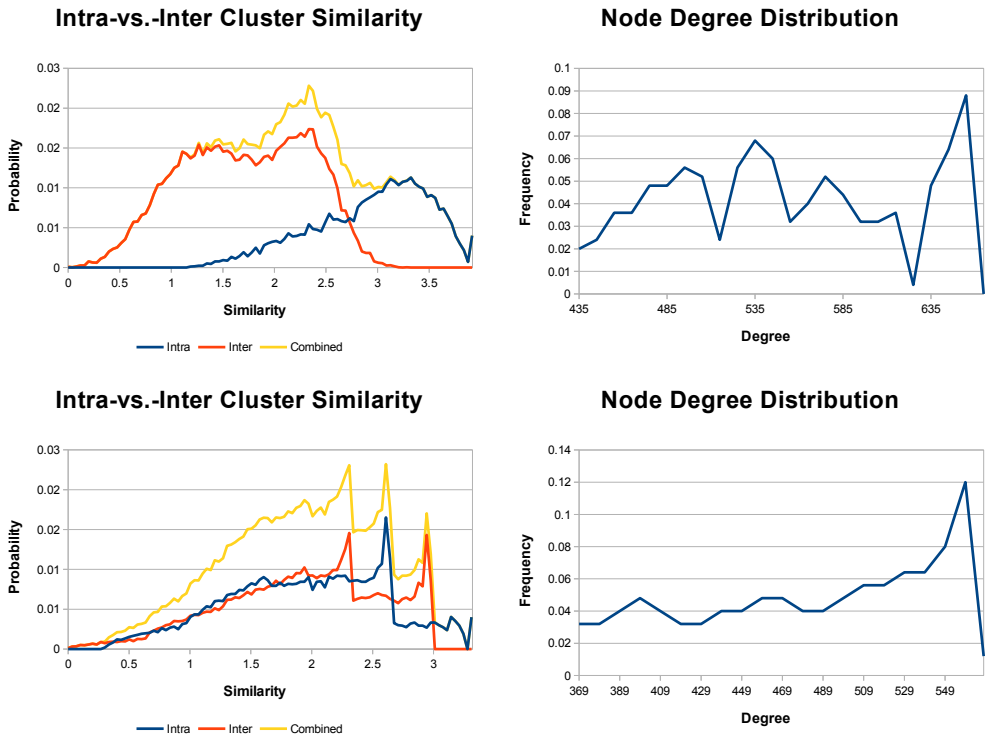


Figure A.1.: This figure displays the statistics calculated for the Cassini dataset in the top part, the Spiral dataset in the lower part. On the left side, the intra and inter cluster similarity is displayed. The overlap between both curves is relatively small which is an indicator of the cluster-ability of the dataset. On the right, the node-degree distribution is displayed. In that case, the node-distribution is the combined weight of all adjacent edges of an object.

Property	Cassini	Spiral
Intra-vs.Inter Cluster Similarity	Figure A.1	Figure A.1
Overlap Intra-vs.-Inter	0.189	0.419
Similarity Distribution	Figure A.1	Figure A.1
Graph Adhesion	248	248
Graph Cohesion	248	248
Graph Min-Cut	110.31	110.55
Clustering Coefficient	1.00	1.00
Node Degree Distribution	Figure A.1	Figure A.1
Graph Density	1.00	1.00
Graph Diversity	0.99	0.99
Matrix Rank	250	250

Table A.1.: Summery of the calculated properties of the Cassini and Spiral dataset.

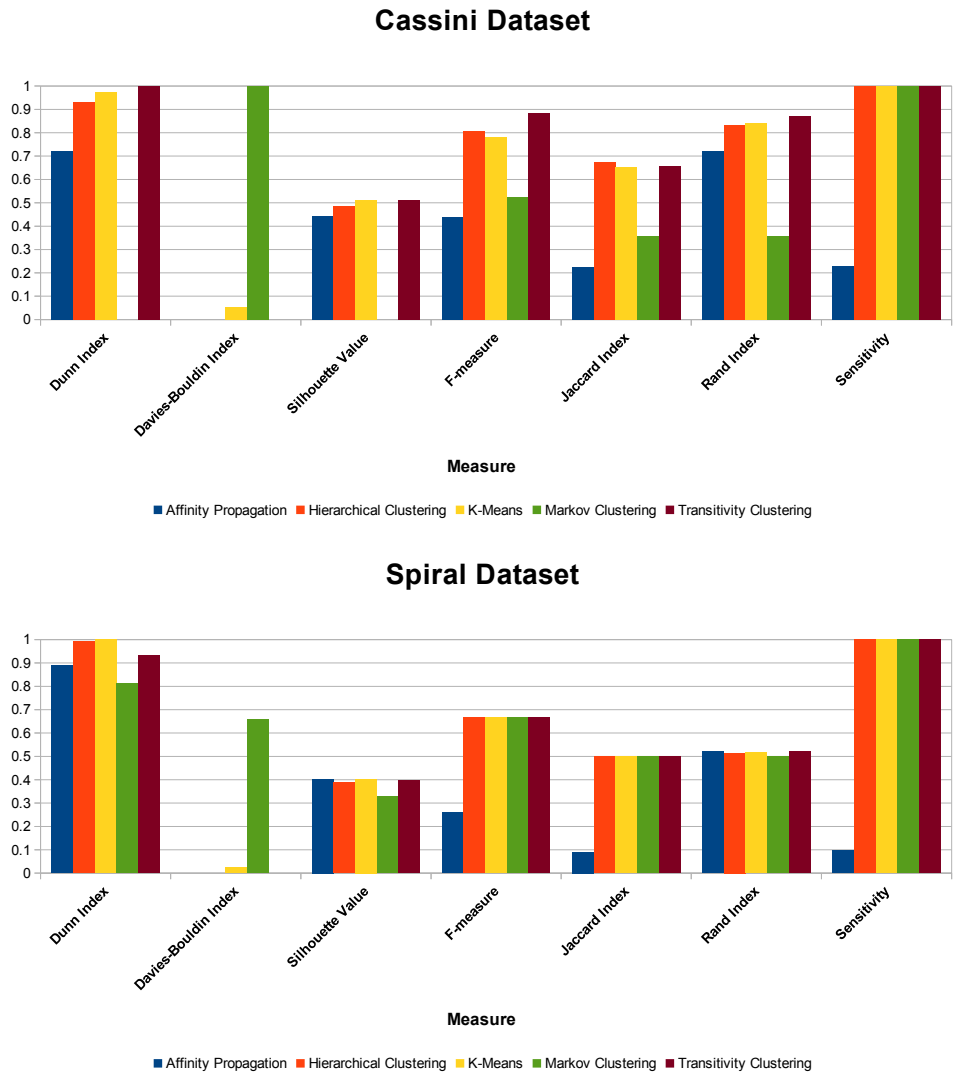


Figure A.2.: This Figure shows the archived cluster quality measures for the Cassini and Spiral dataset. The values for the Dunn Index are scaled to 1 as the Dunn index does not have a bound value range.

B. EHECRegNet

Table B.1.: Findings for the indicated gene of interest. The columns are: "**Pathogen**" - strain name of the pathogen; "**TG**" - the homologous target gene in the pathogen; "**TF**" - the regulating transcription factor of the target gene in the pathogen; "**BS**" - binding site; "**#TG**" - number of target gene regulated by this transcription factor; "**Hom. TF**" - the homologous transcription factor in *E. coli* K-12.

Pathogen	**TG**	**TF**	**BS**	**#TG**	**Hom. TF**
Target gene of Interest: b3025 (gseb)					
O55:H7 str. CB9615	g2583_3746 (qseb)	g2583_3746 (qseb)	acaattacggatt	4	b3025 (qseb)
O111:H- str. 11128	eco111_3848 (qseb)	eco111_3848 (qseb)	acaattacggatt	2	b3025 (qseb)
O26:H11 str. 11368	eco26_4126 (qseb)	eco26_4126 (qseb)	acaattacggatt	2	b3025 (qseb)
UMN026	ecumn_3510 (qseb)	ecumn_3510 (qseb)	acaattacggatt	2	b3025 (qseb)
IAI39	eciai39_3520 (qseb)	eciai39_3520 (qseb)	acaattacggatt	2	b3025 (qseb)
E24377A	ece24377a_3490 (qseb)	ece24377a_3490 (qseb)	acaattacggatt	2	b3025 (qseb)
O157:H7 str. TW14359	ecsp_3999 (qseb)	ecsp_3999 (qseb)	acaattacggatt	2	b3025 (qseb)
55989	ec55989_3441 (qseb)	ec55989_3441 (qseb)	acaattacggatt	2	b3025 (qseb)
O103:H2 str. 12009	eco103_3703 (qseb)	eco103_3703 (qseb)	acaattacggatt	2	b3025 (qseb)
S88	ecs88_3411 (qseb)	ecs88_3411 (qseb)	acaattacggatt	1	b3025 (qseb)
O157:H7 str. EC4115	ech74115_4334 (qseb)	ech74115_4334 (qseb)	acaattacggatt	2	b3025 (qseb)
O157:H7 str. Sakai	ecs3907 ()	ecs3907 ()	acaattacggatt	2	b3025 (qseb)
536	ecp_3112 ()	ecp_3112 ()	acaattacggatt	2	b3025 (qseb)
O127:H6 str. E2348/69	e2348c_3315 (qseb)	e2348c_3315 (qseb)	acaattacggatt	2	b3025 (qseb)
O157:H7 str. EDL933	z4377 (qseb)	z4377 (qseb)	acaattacggatt	2	b3025 (qseb)

Pathogen	TG	TF	BS	#TG	Hom. TF
UTI89	uti89_c3450 (ygix)	uti89_c3450 (ygix)	acaattacggatt	2	b3025 (qseb)
Target gene of Interest: b1892 (flhd)					
O55:H7 str. CB9615	g2583_2345 (flhd)	g2583_4063 (crp)	tgtgtgatctgcatcacacatt	368	b3357 (crp)
		g2583_2759 (rcsb)	taggaaaaatctta	42	b2217 (rcsb)
		g2583_4102 (ompr)	ttttgtaaaataattgtaat	37	b3405 (ompr)
		g2583_0844 (fur)	agattacgattaataaaaa	62	b0683 (fur)
		g2583_2402 (rcsa)	taggaaaaatctta	42	b1951 (rcsa)
		g2583_2826 (lrha)	ttcacatttctgggg	15	b2289 (lrha)
		g2583_2159 (ihfa)	attcattaactca	100	b1712 (ihfa)
		g2583_4247 (gade)	aatccttttagaa	46	b3512 (gade)
O111:H- str. 11128	eco111_2478 (flhd)	eco111_2533 (rcsa)	taggaaaaatctta	9	b1951 (rcsa)
		eco111_0702 (fur)	agattacgattaataaaaa	71	b0683 (fur)
		eco111_2954 (rcsb)	taggaaaaatctta	9	b2217 (rcsb)
		eco111_4326 (gade)	aatccttttagaa	8	b3512 (gade)
		eco111_4167 (crp)	tgtgtgatctgcatcacacatt	306	b3357 (crp)
		eco111_3037 (lrha)	ttcacatttctgggt	6	b2289 (lrha)
		eco111_4214 (ompr)	aaatcttagataagtgtaaa	19	b3405 (ompr)
		eco111_2221 (ihfa)	attcattaactca	143	b1712 (ihfa)
O26:H11 str. 11368	eco26_2744 (flhd)	eco26_2484 (ihfa)	attcattaactca	147	b1712 (ihfa)
		eco26_4493 (ompr)	aaatcttagataagtgtaaa	20	b3405 (ompr)
		eco26_0747 (fur)	agattacgattaataaaaa	79	b0683 (fur)

Pathogen	TG	TF	BS	#TG	Hom. TF
		eco26_2838 (rcsa)	taggaaaaatctta	23	b1951 (rcsa)
		eco26_3144 (rcsb)	taggaaaaatctta	23	b2217 (rcsb)
		eco26_3277 (lrha)	ttcacatttctgggt	6	b2289 (lrha)
		eco26_4446 (crp)	tgtgtgatctgcatcacacatt	301	b3357 (crp)
		eco26_4602 (gade)	aatccttttagaa	22	b3512 (gade)
UMN026	ecumn_2189 (flhd)	ecumn_0768 (fur)	agattacgattaataaaaa	77	b0683 (fur)
		ecumn_3820 (crp)	tgtgtgatctgcatcacacatt	329	b3357 (crp)
		ecumn_2003 (ihfa)	attcattaactca	140	b1712 (ihfa)
		ecumn_2243 (rcsa)	taggaaaaatctta	24	b1951 (rcsa)
		ecumn_4012 (gade)	aatccttttagaa	22	b3512 (gade)
		ecumn_2554 (rcsb)	taggaaaaatctta	24	b2217 (rcsb)
		ecumn_3864 (ompr)	ttttgtaaaataattgtaac	23	b3405 (ompr)
		ecumn_2628 (lrha)	ttcacatttctgggg	6	b2289 (lrha)
IAI39	eciai39_1158 (flhd)	eciai39_4014 (gade)	aatccttttagaa	20	b3512 (gade)
		eciai39_3841 (crp)	tatgtgatctgcatcacacatt	296	b3357 (crp)
		eciai39_1105 (rcsa)	taggaaaaatctta	22	b1951 (rcsa)
		eciai39_0640 (fur)	agattacgattaataaaaa	71	b0683 (fur)
		eciai39_1341 (ihfa)	attcacaaacaaa	138	b1712 (ihfa)
		eciai39_3885 (ompr)	aaatcttagataagtgtaaa	15	b3405 (ompr)
		eciai39_2436 (lrha)	ttcacatttctgggg	1	b2289 (lrha)
		eciai39_2355 (rcsb)	taggaaaaatctta	22	b2217 (rcsb)

Pathogen	TG	TF	BS	#TG	Hom. TF
E24377A	ece24377a_2125 (flhd)	ece24377a_1931 (ihfa)	attcattaactca	141	b1712 (ihfa)
		ece24377a_2582 (lrha)	ttcacatttctgggt	12	b2289 (lrha)
		ece24377a_3998 (gade)	aatccttttagaa	17	b3512 (gade)
		ece24377a_2184 (rcsa)	taggaaaaatctta	22	b1951 (rcsa)
		ece24377a_2517 (rcsb)	taggaaaaatctta	22	b2217 (rcsb)
		ece24377a_3827 (crp)	tgtgtgatctgcatcacacatt	297	b3357 (crp)
		ece24377a_3879 (ompr)	aaatcttagataagtgtaaa	29	b3405 (ompr)
		ece24377a_0711 (fur)	agattacgattaataaaaa	85	b0683 (fur)
O157:H7 str. TW14359	ecsp_2466 (flhd)	ecsp_2555 (rcsa)	taggaaaaatctta	30	b1951 (rcsa)
		ecsp_4355 (ompr)	aaatcttagataagtgtaaa	28	b3405 (ompr)
		ecsp_3097 (rcsb)	taggaaaaatctta	30	b2217 (rcsb)
		ecsp_4501 (gade)	aatccttttagaa	28	b3512 (gade)
		ecsp_4315 (crp)	tgtgtgatctgcatcacacatt	313	b3357 (crp)
		ecsp_2279 (ihfa)	attcattaactca	141	b1712 (ihfa)
		ecsp_3163 (lrha)	ttcacatttctgggg	12	b2289 (lrha)
		ecsp_0731 (fur)	agattacgattaataaaaa	78	b0683 (fur)
55989	ec55989_2071 (flhd)	ec55989_3763 (crp)	tgtgtgatctgtatcacacatt	297	b3357 (crp)
		ec55989_1880 (ihfa)	attcattaactca	133	b1712 (ihfa)
		ec55989_2472 (rcsb)	taggaaaaatctta	16	b2217 (rcsb)
		ec55989_0669 (fur)	agattacgattaataaaaa	71	b0683 (fur)
		ec55989_3955 (gade)	aatccttttagaa	14	b3512 (gade)
		ec55989_2171 (rcsa)	taggaaaaatctta	16	b1951 (rcsa)
		ec55989_2533 (lrha)	ttcacatttctgggg	6	b2289 (lrha)

Pathogen	TG	TF	BS	#TG	Hom. TF
		ec55989_3813 (ompr)	aaatcttagataagtgtaaa	19	b3405 (ompr)
O103:H2 str. 12009	eco103_2154 (flhd)	eco103_2753 (lrha)	ttcacatttctgggt	6	b2289 (lrha)
		eco103_4076 (crp)	tgtgtgatctgcatcacacatt	296	b3357 (crp)
		eco103_4123 (ompr)	aaatcttagataagtgtaaa	21	b3405 (ompr)
		eco103_0679 (fur)	agattacgattaataaaaa	70	b0683 (fur)
		eco103_2693 (rcsb)	taggaaaaatctta	13	b2217 (rcsb)
		eco103_1903 (ihfa)	attcattaactca	152	b1712 (ihfa)
		eco103_2202 (rcsa)	taggaaaaatctta	13	b1951 (rcsa)
		eco103_4240 (gade)	aatccttttagaa	11	b3512 (gade)
S88	ecs88_1949 (flhd)	ecs88_1763 (ihfa)	attcacaaacaaa	139	b1712 (ihfa)
		ecs88_2436 (lrha)	ttcacatttctgggg	6	b2289 (lrha)
		ecs88_2004 (rcsa)	taggaaaaatctta	24	b1951 (rcsa)
		ecs88_3924 (gade)	aatccttttagaa	22	b3512 (gade)
		ecs88_3792 (ompr)	ttttgtaaaataattgtaac	24	b3405 (ompr)
		ecs88_2366 (rcsb)	taggaaaaatctta	24	b2217 (rcsb)
		ecs88_0719 (fur)	agattacgattaataaaaa	78	b0683 (fur)
		ecs88_3748 (crp)	tgtgtgatctgcatcacacatt	279	b3357 (crp)
O157:H7 str. EC4115	ech74115_2631 (flhd)	ech74115_4667 (crp)	tgtgtgatctgcatcacacatt	300	b3357 (crp)
		ech74115_2728 (rcsa)	taggaaaaatctta	27	b1951 (rcsa)
		ech74115_3428 (lrha)	ttcacatttctgggg	6	b2289 (lrha)
		ech74115_4711 (ompr)	aaatcttagataagtgtaaa	20	b3405 (ompr)
		ech74115_4871 (gade)	aatccttttagaa	25	b3512 (gade)

Pathogen	TG	TF	BS	#TG	Hom. TF
		ech74115_2430 (ihfa)	attcattaactca	126	b1712 (ihfa)
		ech74115_0778 (fur)	agattacgattaataaaaa	67	b0683 (fur)
		ech74115_3357 (rcsb)	taggaaaaatctta	27	b2217 (rcsb)
O157:H7 str. Sakai	ecs2602 ()	ecs2690 ()	taggaaaaatctta	30	b1951 (rcsa)
		ecs4247 (ompr)	aaatcttagataagtgtaaa	26	b3405 (ompr)
		ecs3173 ()	ttcacatttctgggg	12	b2289 (lrha)
		ecs0714 (fur)	agattacgattaataaaaa	75	b0683 (fur)
		ecs4208 ()	tgtgtgatctgcatcacacatt	301	b3357 (crp)
		ecs3106 ()	taggaaaaatctta	30	b2217 (rcsb)
		ecs4392 ()	aatccttttagaa	28	b3512 (gade)
		ecs2419 (ihfa)	attcattaactca	141	b1712 (ihfa)
536	ecp_1836 ()	ecp_3448 ()	tatgtgatctgcatcacacatt	267	b3357 (crp)
		ecp_0703 (fur)	agattacgattaataaaaa	72	b0683 (fur)
		ecp_1885 ()	taggaaaaatctta	25	b1951 (rcsa)
		ecp_2260 ()	taggaaaaatctta	25	b2217 (rcsb)
		ecp_3491 (ompr)	aaatcttagataagtgtaaa	24	b3405 (ompr)
		ecp_1660 (ihfa)	attcacaaacaaa	124	b1712 (ihfa)
		ecp_2328 ()	ttcacatttctgggg	7	b2289 (lrha)
		ecp_3610 ()	aatccttttagaa	23	b3512 (gade)
O127:H6 str. E2348/69	e2348c_2015 (flhd)	e2348c_3650 (ompr)	aaatcttagataagtgtaaa	21	b3405 (ompr)
		e2348c_0574 (fur)	agattacgattaataaaaa	71	b0683 (fur)
		e2348c_2065 (rcsa)	taggaaaaatctta	13	b1951 (rcsa)
		e2348c_3754 (gade)	aatccttttagaa	11	b3512 (gade)
		e2348c_2362 (rcsb)	taggaaaaatctta	13	b2217 (rcsb)
		e2348c_3607 (crp)	tgtgtgatctgcatcacacatt	257	b3357 (crp)
		e2348c_1841 (ihfa)	attcacaaacaaa	126	b1712 (ihfa)
		e2348c_2429 (lrha)	ttcacatttctgggg	6	b2289 (lrha)
O157:H7 str. EDL933	z2946 (flhd)	z3549 (lrha)	ttcacatttctgggg	12	b2289 (lrha)
		z2741 (ihfa)	attcattaactca	139	b1712 (ihfa)
		z0831 (fur)	agattacgattaataaaaa	86	b0683 (fur)

Pathogen	TG	TF	BS	#TG	Hom. TF
		z3041 (rcsa)	taggaaaaatctta	30	b1951 (rcsa)
		z4718 (crp)	tgtgtgatctgcatcacacatt	330	b3357 (crp)
		z4925 (yhie)	aatccttttagaa	28	b3512 (gade)
		z4760 (ompr)	ttttgtaaaataattgtaat	27	b3405 (ompr)
		z3476 (rcsb)	taggaaaaatctta	30	b2217 (rcsb)
UTI89	uti89_c2095 (flhd)	uti89_c2570 (lrha)	ttcacatttctgggg	7	b2289 (lrha)
		uti89_c0687 (fur)	agattacgattaataaaaa	66	b0683 (fur)
		uti89_c4043 (yhie)	aatccttttagaa	18	b3512 (gade)
		uti89_c1905 (ihfa)	attcacaaacaaa	134	b1712 (ihfa)
		uti89_c2499 (rcsb)	taggaaaaatctta	20	b2217 (rcsb)
		uti89_c3860 (crp)	tgtgtgatctgcatcacacatt	278	b3357 (crp)
		uti89_c2151 (rcsa)	taggaaaaatctta	20	b1951 (rcsa)
		uti89_c3905 (ompr)	ttttgtaaaataattgtaac	23	b3405 (ompr)
Target gene of Interest: b1880 (flhb)					
O55:H7 str. CB9615	g2583_2332 (flhb)	g2583_4063 (crp)		368	b3357 (crp)
		g2583_2759 (rcsb)		42	b2217 (rcsb)
		g2583_4102 (ompr)		37	b3405 (ompr)
		g2583_0844 (fur)		62	b0683 (fur)
		g2583_2402 (rcsa)		42	b1951 (rcsa)
		g2583_2826 (lrha)		15	b2289 (lrha)
		g2583_2159 (ihfa)		100	b1712 (ihfa)
		g2583_4247 (gade)		46	b3512 (gade)
O111:H- str. 11128	eco111_0253 () eco111_2466 (flhb)	eco111_2478 (flhd)	ggaatgagtttggtaa	42	b1892 (flhd)

Pathogen	TG	TF	BS	#TG	Hom. TF
		eco111_2477 (flhc)	ggaatgagtttggtaa	42	b1891 (flhc)
O26:H11 str. 11368	eco26_2732 (flhb)	eco26_2744 (flhd)	ggaatgagtttggtaa	42	b1892 (flhd)
		eco26_2743 (flhc)	ggaatgagtttggtaa	42	b1891 (flhc)
	eco26_0247 ()				
UMN026	ecumn_2177 (flhb)	ecumn_2189 (flhd)	ggaatgagtttggtaa	41	b1892 (flhd)
		ecumn_2188 (flhc)	ggaatgagtttggtaa	41	b1891 (flhc)
	ecumn_0251 ()				
IAI39	eciai39_1170 (flhb)	eciai39_1158 (flhd)	ggaatgagtttggtaa	14	b1892 (flhd)
E24377A	ece24377a_2112 (flhb)	ece24377a_2124 (flhc)	ggaatgagtttggtaa	37	b1891 (flhc)
		ece24377a_2125 (flhd)	ggaatgagtttggtaa	37	b1892 (flhd)
O157:H7 str. TW14359	ecsp_2454 (flhb)	ecsp_2466 (flhd)	ggaatgagtttggtaa	41	b1892 (flhd)
		ecsp_2465 (flhc)	ggaatgagtttggtaa	41	b1891 (flhc)
55989	ec55989_2059 (flhb)	ec55989_2071 (flhd)	ggaatgagtttggtaa	41	b1892 (flhd)
		ec55989_2070 (flhc)	ggaatgagtttggtaa	41	b1891 (flhc)
O103:H2 str. 12009	eco103_2142 (flhb)	eco103_2154 (flhd)	ggaatgagtttggtaa	42	b1892 (flhd)
		eco103_2153 (flhc)	ggaatgagtttggtaa	42	b1891 (flhc)
S88	ecs88_1938 (flhb)	ecs88_1949 (flhd)	ggaatgagtttggtaa	41	b1892 (flhd)
		ecs88_1948 (flhc)	ggaatgagtttggtaa	41	b1891 (flhc)
O157:H7 str. EC4115	ech74115_2617 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)
O157:H7 str. Sakai	ecs2590 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)

Pathogen	TG	TF	BS	#TG	Hom. TF
536	ecp_1825 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)
O127:H6 str. E2348/69	e2348c_2004 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)
O157:H7 str. EDL933	z2934 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)
UTI89	uti89_c2083 (flhb)	ech74115_2631 (flhd)	ggaatgagtttggtaa	31	b1892 (flhd)
		ech74115_2630 (flhc)	ggaatgagtttggtaa	31	b1891 (flhc)
Target gene of Interest: b1922 (flia)					
O55:H7 str. CB9615	g2583_2373 ()				
O111:H- str. 11128	eco111_2502 (flia)	eco111_2478 (flhd)	taacccccaaataacc	42	b1892 (flhd)
		eco111_5036 (nsrr)	aagatgcagca	72	b4178 (nsrr)
		eco111_2477 (flhc)	taacccccaaataacc	42	b1891 (flhc)
O26:H11 str. 11368	eco26_2814 (flia)	eco26_2744 (flhd)	taacccccaaataacc	42	b1892 (flhd)
		eco26_2743 (flhc)	taacccccaaataacc	42	b1891 (flhc)
		eco26_5344 (nsrr)	aagatgcagca	65	b4178 (nsrr)
UMN026	ecumn_2214 (flia)	ecumn_4711 (nsrr)	ttaacgttaaa	74	b4178 (nsrr)
		ecumn_2189 (flhd)	taacccccaaataacc	41	b1892 (flhd)
		ecumn_2188 (flhc)	taacccccaaataacc	41	b1891 (flhc)
IAI39	eciai39_1133 (flia)	eciai39_1158 (flhd)	taacccccaaataacc	14	b1892 (flhd)
		eciai39_4643 (nsrr)	attgagtatat	59	b4178 (nsrr)

Pathogen	TG	TF	BS	#TG	Hom. TF
E24377A	ece24377a_2156 (flia)	ece24377a_2124 (flhc)	taacccccaaataacc	37	b1891 (flhc)
		ece24377a_4737 (nsrr)	ttaacgttaaa	62	b4178 (nsrr)
		ece24377a_2125 (flhd)	taacccccaaataacc	37	b1892 (flhd)
O157:H7 str. TW14359	ecsp_2526 (flia)	ecsp_2466 (flhd)	taacccccaaataacc	41	b1892 (flhd)
		ecsp_2465 (flhc)	taacccccaaataacc	41	b1891 (flhc)
		ecsp_5278 (nsrr)	ttaacgttaaa	65	b4178 (nsrr)
55989	ec55989_2143 (flia)	ec55989_2071 (flhd)	taacccccaaataacc	41	b1892 (flhd)
		ec55989_4733 (nsrr)	attgagtatat	65	b4178 (nsrr)
		ec55989_2070 (flhc)	taacccccaaataacc	41	b1891 (flhc)
O103:H2 str. 12009	eco103_2178 (flia)	eco103_2154 (flhd)	taacccccaaataacc	42	b1892 (flhd)
		eco103_2153 (flhc)	taacccccaaataacc	42	b1891 (flhc)
		eco103_4971 (nsrr)	attgagtatat	67	b4178 (nsrr)
S88	ecs88_1976 (flia)	ecs88_4764 (nsrr)	ttaacgttaaa	75	b4178 (nsrr)
		ecs88_1949 (flhd)	aatcgcccgattaaaa	41	b1892 (flhd)
		ecs88_1948 (flhc)	aatcgcccgattaaaa	41	b1891 (flhc)
O157:H7 str. EC4115	ech74115_2696 (flia)	ech74115_5694 (nsrr)	ttaacgttaaa	60	b4178 (nsrr)
		ech74115_2631 (flhd)	taacccccaaataacc	31	b1892 (flhd)
		ech74115_2630 (flhc)	taacccccaaataacc	31	b1891 (flhc)
O157:H7 str. Sakai	ecs2661 (flia)	ecs5154 ()	ttaacgttaaa	64	b4178 (nsrr)
		ecs2602 ()	taacccccaaataacc	38	b1892 (flhd)
		ecs2601 ()	taacccccaaataacc	38	b1891 (flhc)
536	ecp_1855 (flia)	ecp_4423 ()	ttaacgttaaa	64	b4178 (nsrr)
		ecp_1836 ()	taacccccaaataacc	36	b1892 (flhd)
		ecp_1835 ()	taacccccaaataacc	36	b1891 (flhc)
O127:H6 str. E2348/69	e2348c_2040 (flia)	e2348c_4501 (nsrr)	ttaacgttaaa	63	b4178 (nsrr)

Pathogen	TG	TF	BS	#TG	Hom. TF
		e2348c_2015 (flhd)	aatcgcccgattaaaa	41	b1892 (flhd)
		e2348c_2014 (flhc)	aatcgcccgattaaaa	41	b1891 (flhc)
O157:H7 str. EDL933	z3012 (flia)	z5785 (yjeb)	ttaacgttaaa	64	b4178 (nsrr)
		z2946 (flhd)	taacccccaaataacc	42	b1892 (flhd)
		z2945 (flhc)	taacccccaaataacc	42	b1891 (flhc)
UTI89	uti89_c2123 (flia)	uti89_c2095 (flhd)	aatcgcccgattaaaa	41	b1892 (flhd)
		uti89_c2094 (flhc)	aatcgcccgattaaaa	41	b1891 (flhc)
		uti89_c4778 (yjeb)	ttaacgttaaa	67	b4178 (nsrr)
Target gene of Interest: b1890 (mota)					
O55:H7 str. CB9615	g2583_2343 (mota)	g2583_4063 (crp)		368	b3357 (crp)
		g2583_2759 (rcsb)		42	b2217 (rcsb)
		g2583_4102 (ompr)		37	b3405 (ompr)
		g2583_0844 (fur)		62	b0683 (fur)
		g2583_2402 (rcsa)		42	b1951 (rcsa)
		g2583_2826 (lrha)		15	b2289 (lrha)
		g2583_2159 (ihfa)		100	b1712 (ihfa)
		g2583_4247 (gade)		46	b3512 (gade)
O111:H- str. 11128	eco111_2476 (mota)	eco111_2533 (rcsa)		9	b1951 (rcsa)
		eco111_0702 (fur)		71	b0683 (fur)
		eco111_2954 (rcsb)		9	b2217 (rcsb)
		eco111_4326 (gade)		8	b3512 (gade)
		eco111_4167 (crp)		306	b3357 (crp)

Pathogen	TG	TF	BS	#TG	Hom. TF
		eco111_3037 (lrha)		6	b2289 (lrha)
		eco111_4214 (ompr)		19	b3405 (ompr)
		eco111_2221 (ihfa)		143	b1712 (ihfa)
	eco111_0293 ()				
O26:H11 str. 11368	eco26_2742 (mota)	eco26_2484 (ihfa)		147	b1712 (ihfa)
		eco26_4493 (ompr)		20	b3405 (ompr)
		eco26_0747 (fur)		79	b0683 (fur)
		eco26_2838 (rcsa)		23	b1951 (rcsa)
		eco26_3144 (rcsb)		23	b2217 (rcsb)
		eco26_3277 (lrha)		6	b2289 (lrha)
		eco26_4446 (crp)		301	b3357 (crp)
		eco26_4602 (gade)		22	b3512 (gade)
	eco26_0287 ()				
UMN026	ecumn_2187 (mota)	ecumn_0768 (fur)		77	b0683 (fur)
		ecumn_3820 (crp)		329	b3357 (crp)
		ecumn_2003 (ihfa)		140	b1712 (ihfa)
		ecumn_2243 (rcsa)		24	b1951 (rcsa)
		ecumn_4012 (gade)		22	b3512 (gade)
		ecumn_2554 (rcsb)		24	b2217 (rcsb)
		ecumn_3864 (ompr)		23	b3405 (ompr)
		ecumn_2628 (lrha)		6	b2289 (lrha)
	ecumn_0292 ()				
IAI39	eciai39_1161 (mota)	eciai39_3083 (cpxr)	agtaaaaagacgtaa	55	b3912 (cpxr)

Pathogen	TG	TF	BS	#TG	Hom. TF
E24377A	ece24377a_2123 (mota)	ece24377a_1931 (ihfa)		141	b1712 (ihfa)
		ece24377a_2582 (lrha)		12	b2289 (lrha)
		ece24377a_3998 (gade)		17	b3512 (gade)
		ece24377a_2184 (rcsa)		22	b1951 (rcsa)
		ece24377a_2517 (rcsb)		22	b2217 (rcsb)
		ece24377a_3827 (crp)		297	b3357 (crp)
		ece24377a_3879 (ompr)		29	b3405 (ompr)
		ece24377a_0711 (fur)		85	b0683 (fur)
O157:H7 str. TW14359	ecsp_2464 (mota)	ecsp_2555 (rcsa)		30	b1951 (rcsa)
		ecsp_4355 (ompr)		28	b3405 (ompr)
		ecsp_3097 (rcsb)		30	b2217 (rcsb)
		ecsp_4501 (gade)		28	b3512 (gade)
		ecsp_4315 (crp)		313	b3357 (crp)
		ecsp_2279 (ihfa)		141	b1712 (ihfa)
		ecsp_3163 (lrha)		12	b2289 (lrha)
		ecsp_0731 (fur)		78	b0683 (fur)
55989	ec55989_2069 (mota)	ec55989_3763 (crp)		297	b3357 (crp)
		ec55989_1880 (ihfa)		133	b1712 (ihfa)
		ec55989_2472 (rcsb)		16	b2217 (rcsb)
		ec55989_0669 (fur)		71	b0683 (fur)
		ec55989_3955 (gade)		14	b3512 (gade)
		ec55989_2171 (rcsa)		16	b1951 (rcsa)
		ec55989_2533 (lrha)		6	b2289 (lrha)

Pathogen	TG	TF	BS	#TG	Hom. TF
		ec55989_3813 (ompr)		19	b3405 (ompr)
O103:H2 str. 12009	eco103_2152 (mota)	eco103_2753 (lrha)		6	b2289 (lrha)
		eco103_4076 (crp)		296	b3357 (crp)
		eco103_4123 (ompr)		21	b3405 (ompr)
		eco103_0679 (fur)		70	b0683 (fur)
		eco103_2693 (rcsb)		13	b2217 (rcsb)
		eco103_1903 (ihfa)		152	b1712 (ihfa)
		eco103_2202 (rcsa)		13	b1951 (rcsa)
		eco103_4240 (gade)		11	b3512 (gade)
S88	ecs88_1947 (mota)	ecs88_1763 (ihfa)		139	b1712 (ihfa)
		ecs88_2436 (lrha)		6	b2289 (lrha)
		ecs88_2004 (rcsa)		24	b1951 (rcsa)
		ecs88_3924 (gade)		22	b3512 (gade)
		ecs88_3792 (ompr)		24	b3405 (ompr)
		ecs88_2366 (rcsb)		24	b2217 (rcsb)
		ecs88_0719 (fur)		78	b0683 (fur)
		ecs88_3748 (crp)		279	b3357 (crp)
O157:H7 str. EC4115	ech74115_2629 (mota)	ech74115_4667 (crp)		300	b3357 (crp)
		ech74115_2728 (rcsa)		27	b1951 (rcsa)
		ech74115_3428 (lrha)		6	b2289 (lrha)
		ech74115_4711 (ompr)		20	b3405 (ompr)
		ech74115_4871 (gade)		25	b3512 (gade)

Pathogen	TG	TF	BS	#TG	Hom. TF
		ech74115_2430 (ihfa)		126	b1712 (ihfa)
		ech74115_0778 (fur)		67	b0683 (fur)
		ech74115_3357 (rcsb)		27	b2217 (rcsb)
O157:H7 str. Sakai	ecs2600 ()	ecs2690 ()		30	b1951 (rcsa)
		ecs4247 (ompr)		26	b3405 (ompr)
		ecs3173 ()		12	b2289 (lrha)
		ecs0714 (fur)		75	b0683 (fur)
		ecs4208 ()		301	b3357 (crp)
		ecs3106 ()		30	b2217 (rcsb)
		ecs4392 ()		28	b3512 (gade)
		ecs2419 (ihfa)		141	b1712 (ihfa)
536	ecp_1834 ()	ecp_3448 ()		267	b3357 (crp)
		ecp_0703 (fur)		72	b0683 (fur)
		ecp_1885 ()		25	b1951 (rcsa)
		ecp_2260 ()		25	b2217 (rcsb)
		ecp_3491 (ompr)		24	b3405 (ompr)
		ecp_1660 (ihfa)		124	b1712 (ihfa)
		ecp_2328 ()		7	b2289 (lrha)
		ecp_3610 ()		23	b3512 (gade)
O127:H6 str. E2348/69	e2348c_2013 (mota)	e2348c_3650 (ompr)		21	b3405 (ompr)
		e2348c_0574 (fur)		71	b0683 (fur)
		e2348c_2065 (rcsa)		13	b1951 (rcsa)
		e2348c_3754 (gade)		11	b3512 (gade)
		e2348c_2362 (rcsb)		13	b2217 (rcsb)
		e2348c_3607 (crp)		257	b3357 (crp)
		e2348c_1841 (ihfa)		126	b1712 (ihfa)
		e2348c_2429 (lrha)		6	b2289 (lrha)
O157:H7 str. EDL933	z2944 (mota)	z3549 (lrha)		12	b2289 (lrha)
		z2741 (ihfa)		139	b1712 (ihfa)
		z0831 (fur)		86	b0683 (fur)

Pathogen	TG	TF	BS	#TG	Hom. TF
		z3041 (rcsa)		30	b1951 (rcsa)
		z4718 (crp)		330	b3357 (crp)
		z4925 (yhie)		28	b3512 (gade)
		z4760 (ompr)		27	b3405 (ompr)
		z3476 (rcsb)		30	b2217 (rcsb)
UTI89	uti89_c2093 (mota)	uti89_c2570 (lrha)		7	b2289 (lrha)
		uti89_c0687 (fur)		66	b0683 (fur)
		uti89_c4043 (yhie)		18	b3512 (gade)
		uti89_c1905 (ihfa)		134	b1712 (ihfa)
		uti89_c2499 (rcsb)		20	b2217 (rcsb)
		uti89_c3860 (crp)		278	b3357 (crp)
		uti89_c2151 (rcsa)		20	b1951 (rcsa)
		uti89_c3905 (ompr)		23	b3405 (ompr)
Target gene of Interest: b0957 (ompa)					
O55:H7 str. CB9615	g2583_1192 ()				
O111:H- str. 11128	eco111_1025 (ompa)	eco111_4167 (crp)	atacggactgcctcaagtgtga	306	b3357 (crp)
O26:H11 str. 11368	eco26_1084 (ompa)	eco26_4446 (crp)	atacggactgcctcaagtgtga	301	b3357 (crp)
UMN026	ecumn_1146 (ompa)	ecumn_3820 (crp)	atacggactgcctcaagtgtga	329	b3357 (crp)
IAI39	eciai39_2190 (ompa)	eciai39_3841 (crp)	atacggactgcctcaagtgtga	296	b3357 (crp)
E24377A	ece24377a_1071 (ompa)	ece24377a_3827 (crp)	atacggactgcctcaagtgtga	297	b3357 (crp)
O157:H7 str. TW14359	ecsp_1063 (ompa)	ecsp_4315 (crp)	atacggactgcctcaagtgtga	313	b3357 (crp)
55989	ec55989_1006 (ompa)	ec55989_3763 (crp)	atacggactgcctcaagtgtga	297	b3357 (crp)
O103:H2 str. 12009	eco103_1003 (ompa)	eco103_4076 (crp)	atacggactgcctcaagtgtga	296	b3357 (crp)

Pathogen	TG	TF	BS	#TG	Hom. TF
S88	ecs88_0978 (ompa)	ecs88_3748 (crp)	atacggactgcctcaagtgtga	279	b3357 (crp)
O157:H7 str. EC4115	ech74115_1121 (ompa)	ech74115_4667 (crp)	atacggactgcctcaagtgtga	300	b3357 (crp)
O157:H7 str. Sakai	ecs1041 ()	ecs4208 ()	atacggactgcctcaagtgtga	301	b3357 (crp)
536	ecp_0962 ()	ecp_3448 ()	atacggactgcctcaagtgtga	267	b3357 (crp)
O127:H6 str. E2348/69	e2348c_0943 (ompa)	e2348c_3607 (crp)	atacggactgcctcaagtgtga	257	b3357 (crp)
O157:H7 str. EDL933	z1307 (ompa)	z4718 (crp)	atacggactgcctcaagtgtga	330	b3357 (crp)
UTI89	uti89_c1022 (ompa)	uti89_c3860 (crp)	atacggactgcctcaagtgtga	278	b3357 (crp)
Target gene of Interest: b3405 (ompr)					
O55:H7 str. CB9615	g2583_4102 (ompr)	g2583_2159 (ihfa)	attcgagaacaaa	100	b1712 (ihfa)
O111:H- str. 11128	eco111_4214 (ompr)	eco111_2221 (ihfa)	attcgagaacaaa	143	b1712 (ihfa)
O26:H11 str. 11368	eco26_4493 (ompr)	eco26_2484 (ihfa)	attcgagaacaaa	147	b1712 (ihfa)
UMN026	ecumn_3864 (ompr)	ecumn_2003 (ihfa)	attcgagaacaaa	140	b1712 (ihfa)
IAI39	eciai39_3885 (ompr)	eciai39_1341 (ihfa)	aatcaataatgtt	138	b1712 (ihfa)
E24377A	ece24377a_3879 (ompr)	ece24377a_1931 (ihfa)	attcgagaacaaa	141	b1712 (ihfa)
O157:H7 str. TW14359	ecsp_4355 (ompr)	ecsp_2279 (ihfa)	attcgagaacaaa	141	b1712 (ihfa)
55989	ec55989_3813 (ompr)	ec55989_1880 (ihfa)	attcgagaacaaa	133	b1712 (ihfa)
O103:H2 str. 12009	eco103_4123 (ompr)	eco103_1903 (ihfa)	attcgagaacaaa	152	b1712 (ihfa)
S88	ecs88_3792 (ompr)	ecs88_1763 (ihfa)	attcgagaacaaa	139	b1712 (ihfa)
O157:H7 str. EC4115	ech74115_4711 (ompr)	ech74115_2430 (ihfa)	attcgagaacaaa	126	b1712 (ihfa)
O157:H7 str. Sakai	ecs4247 (ompr)	ecs2419 (ihfa)	attcgagaacaaa	141	b1712 (ihfa)
536	ecp_3491 (ompr)	ecp_1660 (ihfa)	attcgagaacaaa	124	b1712 (ihfa)
O127:H6 str. E2348/69	e2348c_3650 (ompr)	e2348c_1841 (ihfa)	attcgagaacaaa	126	b1712 (ihfa)

Pathogen	TG	TF	BS	#TG	Hom. TF
O157:H7 str. EDL933	z4760 (ompr)	z2741 (ihfa)	attcgagaacaaa	139	b1712 (ihfa)
UTI89	uti89_c3905 (ompr)	uti89_c1905 (ihfa)	attcgagaacaaa	134	b1712 (ihfa)
Target gene of Interest: b3507 (dctr)					
O55:H7 str. CB9615	g2583_4233 (dctr)	g2583_4227 (arsr)		6	b3501 (arsr)
O111:H- str. 11128	eco111_4321 ()				
O26:H11 str. 11368	eco26_4597 ()				
UMN026	ecumn_3998 ()				
IAI39	eciai39_4000 ()				
E24377A	ece24377a_3993 ()				
O157:H7 str. TW14359	ecsp_4487 ()				
55989	ec55989_3950 ()				
O103:H2 str. 12009	eco103_4235 ()				
S88	ecs88_3911 ()				
O157:H7 str. EC4115	ech74115_4855 ()				
O157:H7 str. Sakai	ecs4378 ()				
536	ecp_3595 ()				
O127:H6 str. E2348/69	e2348c_3741 ()				
O157:H7 str. EDL933	z4909 ()				
UTI89	uti89_c4026 ()				
Target gene of Interest: b3512 (gade)					
O55:H7 str. CB9615	g2583_4247 (gade)	g2583_4247 (gade)		46	b3512 (gade)
		g2583_1406 (phop)		62	b1130 (phop)
O111:H- str. 11128	eco111_4326 (gade)	eco111_4167 (crp)	ttataagaagtctctagtgttt	306	b3357 (crp)
		eco111_4329 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		eco111_4330 (gadx)	gtgtttgaccaataactattg	5	b3516 (gadx)
		eco111_4326 (gade)	taggcgtttactat	8	b3512 (gade)

Pathogen	TG	TF	BS	#TG	Hom. TF
		eco111_1478 (phop)	tatttacaaattgataa	40	b1130 (phop)
O26:H11 str. 11368	eco26_4602 (gade)	eco26_4606 (gadx)	gtgtttgaccaataactattg	4	b3516 (gadx)
		eco26_1664 (phop)	tatttacaaattgataa	40	b1130 (phop)
		eco26_4605 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	2	b3515 (gadw)
		eco26_4602 (gade)	taggcgtttactat	22	b3512 (gade)
		eco26_4446 (crp)	ttataagaagtctctagtgttt	301	b3357 (crp)
UMN026	ecumn_4012 (gade)	ecumn_4017 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		ecumn_4016 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaaca	3	b3515 (gadw)
		ecumn_1374 (phop)	tatttacaaactgttaa	39	b1130 (phop)
		ecumn_3820 (crp)	ttataagaagtctctagtgttt	329	b3357 (crp)
		ecumn_4012 (gade)	taggcgtttactat	22	b3512 (gade)
IAI39	eciai39_4014 (gade)	eciai39_4014 (gade)	taggcgtttactat	20	b3512 (gade)
		eciai39_2007 (phop)	cgttaacttttttgttta	36	b1130 (phop)
		eciai39_3841 (crp)	ttataagaagtctctagtgttt	296	b3357 (crp)
		eciai39_4019 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		eciai39_4018 (gadw)	tttaccatttacaaactgata acaaccaggaattttacttag	3	b3515 (gadw)
E24377A	ece24377a_3998 (gade)	ece24377a_1293 (phop)	cgttaacttttttgttta	36	b1130 (phop)
		ece24377a_4003 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		ece24377a_3998 (gade)	taggcgtttactat	17	b3512 (gade)
		ece24377a_4001 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ece24377a_3827 (crp)	ttataagaagtctctagtgttt	297	b3357 (crp)

Pathogen	TG	TF	BS	#TG	Hom. TF
O157:H7 str. TW14359	ecsp_4501 (gade)	ecsp_4506 (gadx)	gtttactatttacaagctgat	6	b3516 (gadx)
		ecsp_4315 (crp)	ttataagaagtctctagtgttt	313	b3357 (crp)
		ecsp_4504 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ecsp_4501 (gade)	taggcgtttactat	28	b3512 (gade)
		ecsp_1510 (phop)	cgttaacttttgttta	40	b1130 (phop)
55989	ec55989_3955 (gade)	ec55989_3961 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		ec55989_3960 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ec55989_3955 (gade)	taggcgtttactat	14	b3512 (gade)
		ec55989_3763 (crp)	ttataagaagtctctagtgttt	297	b3357 (crp)
		ec55989_1243 (phop)	cgttaacttttgttta	36	b1130 (phop)
O103:H2 str. 12009	eco103_4240 (gade)	eco103_1254 (phop)	tatttacaaattgataa	40	b1130 (phop)
		eco103_4244 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		eco103_4243 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		eco103_4076 (crp)	ttataagaagtctctagtgttt	296	b3357 (crp)
		eco103_4240 (gade)	taggcgtttactat	11	b3512 (gade)
S88	ecs88_3924 (gade)	ecs88_3924 (gade)	taggcgtttactat	22	b3512 (gade)
		ecs88_3929 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		ecs88_3928 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ecs88_1145 (phop)	agttaacctttgttta	41	b1130 (phop)
		ecs88_3748 (crp)	ttataagaagtctctagtgttt	279	b3357 (crp)
O157:H7 str. EC4115	ech74115_4871 (gade)	ech74115_4871 (gade)	taggcgtttactat	25	b3512 (gade)
		ech74115_4667 (crp)	ttataagaagtctctagtgttt	300	b3357 (crp)

Pathogen	TG	TF	BS	#TG	Hom. TF
		ech74115_1591 (phop)	cgttaacttttttgttta	29	b1130 (phop)
		ech74115_4877 (gadx)	gtttactatttacaagctgat	6	b3516 (gadx)
		ech74115_4875 (gadw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
O157:H7 str. Sakai	ecs4392 ()	ecs1602 ()	cgttaacttttttgttta	42	b1130 (phop)
		ecs4208 ()	ttataagaagtctctagtgttt	301	b3357 (crp)
		ecs4396 ()	gtttactatttacaagctgat	6	b3516 (gadx)
		ecs4395 ()	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ecs4392 ()	taggcgtttactat	28	b3512 (gade)
536	ecp_3610 ()	ecp_3615 ()	gtgtttgaccaataactattg	6	b3516 (gadx)
		ecp_3448 ()	ttataagaagtctctagtgttt	267	b3357 (crp)
		ecp_3614 ()	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		ecp_3610 ()	taggcgtttactat	23	b3512 (gade)
		ecp_1125 ()	agttaacctttttgttta	36	b1130 (phop)
O127:H6 str. E2348/69	e2348c_3754 (gade)	e2348c_3758 (gadx)	gtgtttgaccaataactattg	6	b3516 (gadx)
		e2348c_3757 (gadw)	atgggcggttaaataagcaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		e2348c_3754 (gade)	taggcgtttactat	11	b3512 (gade)
		e2348c_1271 (phop)	agttaacctttttgttta	44	b1130 (phop)
		e2348c_3607 (crp)	ttataagaagtctctagtgttt	257	b3357 (crp)
O157:H7 str. EDL933	z4925 (yhie)	z4718 (crp)	ttataagaagtctctagtgttt	330	b3357 (crp)
		z1859 (phop)	cgttaacttttttgttta	48	b1130 (phop)
		z4929 (yhix)	gtttactatttacaagctgat	6	b3516 (gadx)
		z4928 (yhiw)	atgggcggttaaataagtaa tccgggttcatttttttgcaac	3	b3515 (gadw)
		z4925 (yhie)	taggcgtttactat	28	b3512 (gade)
UTI89	uti89_c4043 (yhie)	uti89_c4048 (yhix)	tcaaacattatcatggctgat	6	b3516 (gadx)
		uti89_c4047 (yhiw)	attagccatttcaaacattat catggctgatattttccgtgg	3	b3515 (gadw)
		uti89_c1259 (phop)	agttaacctttttgttta	40	b1130 (phop)

Pathogen	TG	TF	BS	#TG	Hom. TF
		uti89_c4043 (yhie)	taggcgtttactat	18	b3512 (gade)
		uti89_c3860 (crp)	ttataagaagtctctagtgttt	278	b3357 (crp)

C. Actinobacterial Dataset

C.1. Complete Dataset

Table C.1.: List of all actinobacteria used in the Actino dataset. The column "Path." (Pathogenicity) denotes if the species is a human pathogen (HP), an animal pathogen (AP), an opportunistic pathogen (OP) or a non-pathogen (NP). The last column names the associated disease, if available.

Species	Path.	Disease
Corynebacterium accolens ATCC 49725	OP	NA
Corynebacterium accolens ATCC 49726	OP	NA
Corynebacterium ammoniagenes DSM 20306	OP	NA
Corynebacterium amycolatum SK46	HP	Endocarditis Sepsis
Corynebacterium aurimucosum ATCC 700975	OP	Opportunistic infections
Corynebacterium diphtheriae NCTC 13129	HP	Diphtheria
Corynebacterium efficiens YS-314	NP	None
Corynebacterium genitalium ATCC 33030	OP	NA
Corynebacterium glucuronolyticum ATCC 51866	OP	NA
Corynebacterium glucuronolyticum ATCC 51867	OP	NA
Corynebacterium glutamicum ATCC 13032	NP	None
Corynebacterium glutamicum R	NP	NA
Corynebacterium jeikeium ATCC 43734	OP	NA
Corynebacterium jeikeium K411	OP	Nocosomial infections
Corynebacterium kroppenstedtii DSM 44385	HP	NA
Corynebacterium lipophiloflavum DSM 44291	HP	NA
Corynebacterium matruchotii ATCC 14266	OP	Oral infection
Corynebacterium matruchotii ATCC 33806	OP	NA
Corynebacterium pseudogenitalium ATCC 33035	OP	NA
Corynebacterium pseudotuberculosis 1002	AP	caseous lymphadenitis (CLA)
Corynebacterium pseudotuberculosis C231	AP	Lymphadenitis
Corynebacterium pseudotuberculosis FRC41	AP	NA
Corynebacterium pseudotuberculosis I19	AP	NA
Corynebacterium resistens DSM 45100	OP	NA
Corynebacterium striatum ATCC 6940	OP	NA
Corynebacterium urealyticum DSM 7109	OP	Urinary tract infection
Corynebacterium variabile DSM 44702	NP	NA

Species	Path.	Disease
Mycobacterium avium 104	OP	Tuberculosis type pulmonary infections
Mycobacterium avium subsp. avium ATCC 25291	OP	Tuberculosis type pulmonary infections
Mycobacterium avium subsp. paratuberculosis K-10	AP	Paratuberculosis
Mycobacterium bovis AF2122/97	AP	Tuberculosis in cattle
Mycobacterium bovis BCG str. Pasteur 1173P2	AP	Bovine tuberculosis
Mycobacterium bovis BCG str. Tokyo 172	AP	NA
Mycobacterium gilvum PYR-GCK	NP	NA
Mycobacterium intracellulare ATCC 13950	HP	Tuberculosis-like disease
Mycobacterium kansasii ATCC 12478	OP	Tuberculosis type pulmonary infections
Mycobacterium leprae Br4923	HP	Leprosy
Mycobacterium leprae TN	HP	Leprosy
Mycobacterium marinum M	OP	Tuberculosis-like infection in fish Skin/Infection arthritis in humans
Mycobacterium parascrofulaceum ATCC BAA-614	OP	NA
Mycobacterium smegmatis str. MC2 155	OP	Soft tissue lesions
Mycobacterium sp. JLS	NP	None
Mycobacterium sp. KMS	NP	None
Mycobacterium sp. MCS	NP	None
Mycobacterium sp. Spyr1	NP	NA
Mycobacterium tuberculosis '98-R604 INH-RIF-EM'	HP	Tuberculosis
Mycobacterium tuberculosis 02_1987	HP	Tuberculosis
Mycobacterium tuberculosis 210	HP	Tuberculosis
Mycobacterium tuberculosis 94_M4241A	HP	Tuberculosis
Mycobacterium tuberculosis C	HP	Tuberculosis
Mycobacterium tuberculosis CDC1551	HP	Tuberculosis
Mycobacterium tuberculosis CPHL_A	HP	NA
Mycobacterium tuberculosis EAS054	HP	Tuberculosis
Mycobacterium tuberculosis F11	HP	Tuberculosis
Mycobacterium tuberculosis GM 1503	HP	NA
Mycobacterium tuberculosis H37Ra	HP	Tuberculosis (attenuated)
Mycobacterium tuberculosis H37Rv	HP	Tuberculosis
Mycobacterium tuberculosis K85	HP	NA
Mycobacterium tuberculosis KZN 1435	HP	Tuberculosis
Mycobacterium tuberculosis KZN 4207	HP	Tuberculosis
Mycobacterium tuberculosis KZN 605	HP	Tuberculosis
Mycobacterium tuberculosis KZN R506	HP	NA
Mycobacterium tuberculosis KZN V2475	HP	NA
Mycobacterium tuberculosis str. Haarlem	HP	Tuberculosis

Species	Path.	Disease
Mycobacterium tuberculosis SUMu001	HP	Tuberculosis
Mycobacterium tuberculosis SUMu002	HP	Tuberculosis
Mycobacterium tuberculosis SUMu003	HP	Tuberculosis
Mycobacterium tuberculosis SUMu004	HP	Tuberculosis
Mycobacterium tuberculosis SUMu005	HP	Tuberculosis
Mycobacterium tuberculosis SUMu006	HP	Tuberculosis
Mycobacterium tuberculosis SUMu007	HP	Tuberculosis
Mycobacterium tuberculosis SUMu008	HP	Tuberculosis
Mycobacterium tuberculosis SUMu009	HP	Tuberculosis
Mycobacterium tuberculosis SUMu010	HP	Tuberculosis
Mycobacterium tuberculosis SUMu011	HP	Tuberculosis
Mycobacterium tuberculosis SUMu012	HP	Tuberculosis
Mycobacterium tuberculosis T17	HP	NA
Mycobacterium tuberculosis T46	HP	NA
Mycobacterium tuberculosis T85	HP	Tuberculosis
Mycobacterium tuberculosis T92	HP	Tuberculosis
Mycobacterium ulcerans Agy99	HP	Buruli ulcer
Mycobacterium vanbaalenii PYR-1	NP	NA
Nocardia farcinica IFM 10152	HP	Nocardiosis
Rhodococcus equi 103S	AP	Pneumonia
Rhodococcus equi ATCC 33707	AP	Pneumonia
Rhodococcus erythropolis PR4	OP	NA
Rhodococcus erythropolis SK121	OP	NA
Rhodococcus jostii RHA1	NP	Cocci
Rhodococcus opacus B4	NP	NA

C.2. Additional Proteobacterial Dataset

Table C.2.: List of all proteobacteria used in this study. The selection within each group was based on the number of registered projects in NCBI. Only one species was allowed per family.

Organism	Subgroup
Brucella abortus A13334	Alphaproteobacteria
Rhizobium leguminosarum bv. viciae 3841	Alphaproteobacteria
Sinorhizobium meliloti 1021	Alphaproteobacteria
Bradyrhizobium sp. BTAi1	Alphaproteobacteria
Rickettsia rickettsii str. Iowa	Alphaproteobacteria
Anaplasma marginale str. Florida	Alphaproteobacteria
Acetobacter pasteurianus IFO 3283-01	Alphaproteobacteria

Organism	**Subgroup**
Agrobacterium radiobacter K84	Alphaproteobacteria
Zymomonas mobilis subsp. mobilis ZM4	Alphaproteobacteria
Rhodobacter sphaeroides 2.4.1	Alphaproteobacteria
Neisseria meningitidis M01-240149	Betaproteobacteria
Burkholderia sp. 383	Betaproteobacteria
Ralstonia solanacearum GMI1000	Betaproteobacteria
Bordetella pertussis CS	Betaproteobacteria
Acidovorax sp. JS42	Betaproteobacteria
Comamonas testosteroni CNB-2	Betaproteobacteria
Achromobacter xylosoxidans A8	Betaproteobacteria
Cupriavidus necator N-1	Betaproteobacteria
Taylorella equigenitalis MCE9	Betaproteobacteria
Delftia acidovorans SPH-1	Betaproteobacteria
Escherichia coli str. K-12 substr. MG1655	Gammaproteobacteria
Salmonella enterica subsp. enterica serovar Paratyphi A str. ATCC 9150	Gammaproteobacteria
Acinetobacter baumannii 1656-2	Gammaproteobacteria
Vibrio cholerae O395	Gammaproteobacteria
Yersinia pestis CO92	Gammaproteobacteria
Pseudomonas syringae pv. syringae B728a	Gammaproteobacteria
Xanthomonas axonopodis pv. citri str. 306	Gammaproteobacteria
Klebsiella pneumoniae 342	Gammaproteobacteria
Francisella tularensis subsp. tularensis NE061598	Gammaproteobacteria
Shigella flexneri 2002017	Gammaproteobacteria
Helicobacter pylori 26695	delta/epsilon subdivisions
Campylobacter jejuni RM1221	delta/epsilon subdivisions
Desulfovibrio vulgaris RCH1	delta/epsilon subdivisions
Arcobacter butzleri RM4018	delta/epsilon subdivisions
Geobacter sp. M18	delta/epsilon subdivisions
Stigmatella aurantiaca DW4/3-1	delta/epsilon subdivisions
Myxococcus xanthus DK 1622	delta/epsilon subdivisions
Bacteriovorax marinus SJ	delta/epsilon subdivisions
Anaeromyxobacter sp. K	delta/epsilon subdivisions
Sulfurovum sp. NBC37-1	delta/epsilon subdivisions

C.3. Mapping with the Ortholog Matrix Project

Organism	#Proteins	#Matches
Corynebacterium aurimucosum ATCC 700975	2531	2504
Corynebacterium diphtheriae NCTC 13129	2272	2242
Corynebacterium efficiens YS 314	2938	2824
Corynebacterium glutamicum R	3052	3043
Corynebacterium jeikeium K411	2104	2018
Corynebacterium kroppenstedtii DSM 44385	2018	2018
Corynebacterium pseudotuberculosis C231	2091	1948
Corynebacterium pseudotuberculosis FRC41	2110	2110
Corynebacterium urealyticum DSM 7109	2022	2008
Mycobacterium avium 104	5120	4473
Mycobacterium bovis AF2122 97	3918	3893
Mycobacterium bovis BCG Pasteur 1173P2	3949	3884
Mycobacterium bovis BCG Tokyo 172	3944	3898
Mycobacterium gilvum PYR GCK	5241	5154
Mycobacterium JLS	5739	5693
Mycobacterium KMS	5460	5338
Mycobacterium leprae Br4923	1604	1591
Mycobacterium leprae TN	1605	1582
Mycobacterium marinum M	5423	5388
Mycobacterium MCS	5391	5359
Mycobacterium smegmatis MC2 155	6717	5873
Mycobacterium tuberculosis H37Ra	4034	3915
Mycobacterium tuberculosis H37Rv	4003	3753
Mycobacterium tuberculosis KZN 1435	4059	4017
Mycobacterium ulcerans Agy99	4160	4113
Mycobacterium vanbaalenii PYR 1	5979	5899
Nocardia farcinica IFM 10152	5681	5656
Rhodococcus equi 103S	4512	4492
Rhodococcus erythropolis PR4	6030	6020
Rhodococcus opacus B4	7246	7239

Table C.3.: Summary of the mapping between OMA and the actinobacteria used in our study. The column "#Proteins" gives the number of proteins of the corresponding species and "#matches" denotes the number of successfully matched IDs and sequences.