INVESTIGATING THE k-NEAREST NEIGHBORS RESOLUTION ALGORITHMS FOR PYROPRINTS AND CLUSTERING FOR BACTERIAL STRAINS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Jeffrey D. McGovern

March 2016

COMMITTEE MEMBERSHIP

TITLE:	Investigating The k-Nearest Neighbors Resolution Algorithms for Pyroprints and Clustering for Bacterial Strains
AUTHOR:	Jeffrey D. McGovern
DATE SUBMITTED:	March 2016
COMMITTEE CHAIR:	Alexander Dekhtyar, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Chris Kitts, Ph.D. Professor of Biological Sciences
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D. Professor of Computer Science

ABSTRACT

Investigating The k-Nearest Neighbors Resolution Algorithms for Pyroprints and Clustering for Bacterial Strains

Jeffrey D. McGovern

Your abstract goes in here

ACKNOWLEDGMENTS

Thanks to:

- Andrew Guenther, for uploading this template

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

[1] Recreated from [30]

LIST OF ALGORITHMS

Chapter 1

INTRODUCTION

Fecal contamination in public water sources is an issue that health officials and city and county governments must frequently combat. Pathogens present in fecal matter pose severe health risks to humans and pets and the decomposition of fecal coliform bacteria can upset the balance of aquatic ecosystems by depleting dissolved oxygen to low enough levels that it may kill other species in the water. Such severe threats to the health of humans, pets, and the local ecosystem motivates public health officials to take action in order to mitigate its consequences. Often times, no one observes the cause of the fecal contamination, but rising levels of fecal coliform bacteria indicate that fecal contamination has occurred. In these situations, usually the only course of action that natural resource managers have is to simply restrict public access until contamination levels reach an acceptable level, which may not prevent further contamination. Identifying the source of fecal contamination in water supplies is an important initial step to prevent further contamination.

Microbial Source Tracking (MST) is the field of research that aims to discover the host-species that microbial lifeforms originate from and aids the process of sourcing fecal contamination. Microbes thrive inside the gut of animals, as well as in masses of plant matter, and routinely make their way into the environment via fecal matter deposition. Biologists conjecture that strains of microbes or bacteria present in fecal matter, called fecal indicator bacteria (FIB), remain relatively unique to the species of the host they originated from. A strain of a species of microbe is a subtype of that species where the microbes in that strain are closely related in some meaningful way. How researchers specifically define strains often differs, since each definition of a strain depends on the characterization of the microbe in question and the methods used to

derive such characterizations. Typically, the objective is to discover which microbes of a bacterial isolate came from the same parent microbe [35]. A strain, then, can be thought of as consisting of individuals descending from the same individual to generate a "group" or "family." Researchers put significant effort into choosing the relevant microbes and appropriately characterizing them in order to discover which strains tend to belong to which species.

A common method of MST known as library-based MST involves collecting fecal matter from a known host-species, culturing isolates of the relevant microbes in the fecal matter, and building a digital representation of the collected isolates for storage into a database and analysis. Storing an appropriate digital representation allows researchers to perform rigorous analysis and comparison between FIB isolates collected from different host-animals and host-species, as well as isolates collected from the same host-animal, but at different times. The data inserted into such a database may range from collection metadata about the microbiome, to a specific microbe characterization, or to any other useful set of metrics that can appropriately profile an entry [57].

In this way, researchers build a "library" of known-host-species isolates. Using this library, researchers can take an environmental sample with FIB from an unknown source, process the microbial isolates using the same procedure and the known-host-species isolates in the library, and compare the strain representation of the environmental sample to those in the library to find any close matches. Since the researchers know the host-species of the isolates in the library, they can make a reasonable determination of the source of the isolates in the environmental sample. The methods used to compare isolates and make assertions depends entirely upon the FIB, their method of collection, and their digital representation.

Library-based MST is usually only effective within the region in which the known-

host-species isolates came from, making it difficult to build a "one size fits all" library. Companies exist that can, for fees in the ballpark of $100, attempt to determine the host-species of a provided sample and while these companies exist nationwide, they are few in number and usually cannot build a representative sampling of every region for accurate host-species determination. Additionally, when investigating an incident of fecal contamination, investigators want to send out multiple samples to build reliable evidence for a determination of the source. As a result, the cost of outsourcing becomes too prohibitive and determinations too inaccurate for it to be an option. Thus, there exists a need for a cost-effective and accurate method of MST in order to properly tackle the problem of preventing fecal contamination in water supplies.

In 2009, the California Polytechnic State University San Luis Obispo (Cal Poly) Biology and Computer Science Departments built The Cal Poly Library of Pyroprints (CPLOP) [66], a database of *Escherichia coli* (*E. coli*) isolate fingerprints, called pyroprints. Students collect fecal samples from a variety of host-species from the San Luis Obispo area and build the pyroprints using a low cost DNA sequencing method called pyrosequencing on two intergenic regions of an *E. coli* isolate. Building pyroprints ends up costing roughly two orders of magnitude less than outsourcing samples, cutting the cost of building an effective MST library by as much as 60% [9]. It is through CPLOP that Cal Poly researchers hope to better understand bacterial strains, how to differentiate between them, and provide a cost-effective MST methodology.

In order to be an effective MST fingerprinting method, pyroprinting must contain information that allows for the accurate discrimination between closely related strains of *E. coli* bacteria. Internal Transcribed Spacers (*ITS*) in bacteria are regions of DNA that do not contain instruction for building proteins and thus have high variability, since variability across generations of bacteria does not affect the survivability of

the microbe. Because of this high variability, researchers can use these regions to differentiate between strains of the same species of microbe. *E. coli* isolate pyroprints stored in CPLOP represent the polymerase chain reaction (PCR)-amplified regions of DNA between the *16S* and *23S* genes and *23S* and *5S* genes, referred to as *ITS-1* and *ITS-2* respectively. *ITS-1* and *ITS-2*, along with the entire *E. coli* genome, repeat seven times, giving us seven highly variable regions for each *ITS*. Any offspring inherit mostly accurate copies[1] of the *ITS* regions of the parent microbe, encoding the notion of a "group" or "family" and allowing researchers to use them to differentiate between strains [67]. By building pyroprints out of these regions, CPLOP researchers hope to gain a reproducible notion of an *E. coli* strain that they can use for MST.

A pyroprint is a vector comprised of the peak heights of pyrosequences of multiple copies of a repeated region of DNA. By dispensing a series of nucleotides at specific times and observing the resulting light emitted, CPLOP researchers can build a fingerprint of that DNA sequence. In traditional pyrosequencing, the DNA sequenced is an amplified version of a single sequence of DNA, allowing researchers to reconstruct the exact sequence of nucleotides that make up the DNA. Since CPLOP researchers pyroprint segments of DNA that repeat but are highly variable, researchers cannot reconstruct the exact sequences of the *ITS* sequences. Alternatively, CPLOP contains a pyroprint that represents the random variability in the entire genome of that particular *E. coli* isolate. Previous work in [64] optimized the pyroprinting process, including the dispensation sequence and peak height determination, for each *ITS* to best delineate between different strains of *E. coli* using the Pearson correlation coefficient to compare pyroprints.

The Pearson correlation coefficient ρ normalizes the covariance of two vectors by the standard deviation of each, providing a notion of relative co-variability between

[1]Some variation may occur, but researchers assume it is small for immediately related microbes and large for distantly related microbes.

the vectors that remains invariant of noise and scaling — a core reason why CPLOP researchers use it to compare pyroprints. In order to compare two *E. coli* isolates in CPLOP, researchers must separately compare the *ITS-1* pyroprints to each other and the *ITS-2* pyroprints to each other using Pearson correlation coefficient. It is meaningless to compare different *ITS* to each other since they represent entirely different sections of DNA that have been obtained through a different sequence of dispensations. This effectively gives us two comparison metrics between isolates: the Pearson correlation coefficient between two *ITS-1* and the Pearson correlation coefficient pyroprints between two *ITS-2* pyroprints — $\rho_{ITS\text{-}1}$ and $\rho_{ITS\text{-}2}$. Using these values, CPLOP researchers can rigorously define the notion of a strain.

CPLOP supports numerous research projects, ranging from longitudinal studies of a host-animal to large studies of one or more host-species, in order to understand the evolution and transmission of *E. coli* strains and verify that pyroprinting provides an accurate representation of *E. coli* strains. Previous work on CPLOP include formation and validation of the pyroprinting process, exploration of the evolution and transference of *E. coli* strains within and between host-animal and host-species, and new algorithms designed specifically for CPLOP to better understand its data.

Much of the work done so far using CPLOP has been exploring the composition, evolution, and transference of strains among host-animals and host-species. While part of this is to validate the MST methodology that leverages CPLOP data, researchers gain a large amount of insight into how *E. coli* strains get into and evolve in fecal matter by using pyroprints to rigorously study changes. Clustering methods become very useful in this case, owing their effectiveness to the notion of a strain being similar to a "group" or "family" of a closely related subtype of a species of microbe.

Two pieces of previous work, [40, 41] and [30], worked toward building clustering

algorithms that can provide meaningful insight into the *E. coli* isolates in CPLOP. The former, *OhClust!*, is an agglomerative clustering algorithm where a biologist-provided metadata-ontology guides the agglomeration. The latter, by Eric Johnson, is a density-based clustering algorithm — DBSCAN — optimized by a fast range query for nearby isolates.

While *OhClust!* takes advantage of all of the information available in CPLOP, DBSCAN encodes our notion of a strain the closest and allows for strain discovery without needing to guess what ontology provides the best insight. Moreover, the range query optimizations made in [30] allow for efficient, low-memory querying of isolates while still encoding the notion of Pearson correlation coefficient between isolates, an improvement over *OhClust!*'s need to precompute and store distances in order to mitigate the consequences of agglomerative clustering's need for a high number of distance computations. It may be that by preclustering isolates, we can speed up the computation of k-RAP.

In a nutshell, DBSCAN uses a distance metric, a minimum neighbors value `MinPts`, and an ε range to categorize data points as one of three types: core point, border point, or noise. A core point is a point that has at least `MinPts` data points within ε of it. A border point is a point that is within ε of a core point, but that does not have `MinPts` points within ε of it. Every other point is noise. The algorithm then defines a cluster as a group of neighboring core points with their associated border points.

In [37], we constructed the notion of a bacterial strain purely from the clusters produced by DBSCAN — i.e. we defined bacterial strains to be the clusters produced by DBSCAN. We studied the cluster purity — the proportion of isolates in a cluster that are of the same species — of the entire clustering at different `MinPts` values. In doing so, we observed the presence of so-called transient *E. coli* strains — strains of

E. coli that show up in many different host-species — that tend to confound MST. More importantly, it showed that CPLOP has relatively few of these transient strains and a large number of pure strains.

While the original purpose of CPLOP was to support MST and some manual MST studies have been conducted, little research has been done on building an automated MST method. Most studies performed with CPLOP focused on validating and exploring the various biological features captured by the pyroprinting process and the comparison metric used to compare pyroprints, the Pearson correlation coefficient. Building objective, repeatable classification metrics that use the data in CPLOP to assist MST can help biologists inform investigators of a possible source that caused, or is causing, fecal contamination.

As a first step towards building an effective classification technique, we chose to use the k-Nearest Neighbors (k-NN) classification algorithm on CPLOP to measure how accurately we can classify samples that we know the host-species of. k-NN classifies an unknown-class datum by querying a library of known data — each datum has a class, or classification — for "nearby" data; sorts the list by nearness, limiting it to k many "neighbor" data points; and classifies from this "k-nearest neighbors" list by picking the most plural classification present among the neighbors, breaking ties by average position.

A somewhat unique obstacle arises with the *E. coli* isolates in CPLOP: in order to compare isolates, we must use two different comparison metrics — $\rho_{ITS\text{-}1}$ and $\rho_{ITS\text{-}2}$. For k-NN on CPLOP *E. coli* isolates, this means that we produce two k-nearest neighbors lists that we must classify from. Resolving multiple k-NN lists can be useful for any data that has multiple meaningful-yet-exclusive ways to compare one datum to another. Biologists using k-NN will likely want to restrict the list further than k, since their definition of a strain relies heavily on bounding the Pearson correlation

coefficient between two isolates for both *ITS* — so we also add an α threshold to further limit the involved k-NN lists.

The four resolution algorithms, called the k-NN Resolution Algorithms for Pyroprints (k-RAP) and previously published in [36], are termed: Meanwise Resolution, Resolution by Winner, Resolution by Union, and Resolution by Intersection. Meanwise Resolution takes the average of the comparison value to form a single k-NN list. Resolution by Winner finds the most plural classification in each k-NN list and picks the classification with the most instances of that class in its list. Resolution by Union combines each k-NN list into a single set — performing effectively a union on all of the k-NN lists — and finds the most plural classification in the resulting set, breaking ties by average original position. Resolution by Intersection forms a new set that is exactly the isolates that appear in every k-NN list — effectively performing an intersection at the isolate level — expanding both lists and adding to the set until the set itself is of size k and choosing the most plural class of the new set.

Investigating k-RAP in [36] showed us that classification accuracy for the entire database stayed well above 50% with most of the resolution algorithms. Precision and recall for well-represented host-species also stayed safely above 0.30, which is far better than random and notably better than our outsourced baseline. Underrepresented species predictably performed poorly in classification. Furthermore, α thresholding noticeably improved performance on some resolution algorithms, causing one to perform better than the others with α, but worse without.

In this thesis, we investigate further the work done in [36] and [37] in depth and consider whether combining the two is useful for performing MST. Namely, the contributions of this paper are the following:

- k-NN Resolution Algorithms for Pyroprints: Modifications to the k-NN classification algorithm that can resolve multiple comparison metrics

- A modification to k-NN that adds α thresholding to further restrict the individual k-NN lists
- An empirical study measuring the accuracy of identifying the host-species for the *E. coli* isolates stored in CPLOP, investigating how values of k and α affect the accuracy with each resolution metric
- Revisions to work done in [36]
- An investigation of the efficient density-based clustering algorithm in [30] that is scalable and meaningfully encodes the comparison-metric used in CPLOP
- A set of validation measures for clustering bacterial isolates into strains
- An evaluation of our strain discovery procedure based on the defined set of measures

The rest of this thesis is organized as follows: Chapter 2 provides an overview of relevant work in the field of MST and an introduction to work done using CPLOP; Chapter 3 details CPLOP and the background necessary to understand the algorithms presented; Chapter 4 describes k-RAP and the use of DBSCAN as a clustering method for bacterial strains; Chapter 5 gives an overview of the structure of the code and how to use it; Chapter 6 defines the evaluation criteria that the algorithms are judged by and motivation for their use; Chapter 7 discusses the results of the investigation; and Chapter 8 concludes, offering suggestions for future work.

Chapter 2

RELATED WORK

Existing Microbial Source Tracking (MST) methodologies require a fecal indicator bacteria (FIB) fingerprinting method that allows for strain discrimination and a method of classification. Early work in MST [8] worked by measuring the ratio of fecal coliform bacteria to streptococci ratios, which fell out of use due to the "widely varying survival rates of the bacterial groups in the environment" [60]. In order to effectively use FIB for MST, researchers had to develop new methods to fingerprint and classify them with the appropriate host-species or find related strains.

Fingerprinting FIB usually falls into two categories: phenotyping and genotyping. Phenotypic methods of fingerprinting usually involve "morphology of colonies on various culture media, biochemical tests, serology, killer toxin susceptibility, pathogenicity, and antibiotic susceptibility," none of which allows researchers to reliably distinguish between closely related strains [35]. Genetic fingerprinting — genotyping — "has become widely used ... due to its high resolution" [35] and many methods exist that allow for effective discrimination [61, 60].

Classification methods use a variety of statistical measures to make determinations to either related strains or host-species, but most fall into library-dependent and library-independent. Library-independent MST searches for the presence of certain microbes in fecal matter or contaminated water. The presence of certain microbes can indicate what host-species may have deposited the fecal matter. Unfortunately, this method relies on prior knowledge of the types of microbes that may occur in the types of potential host-species[1], limiting the effectiveness of host-species determination [60].

[1]Often times, these methods can only detect whether the fecal content came from a human and maybe some domestic animal species [60].

Library dependent techniques work by building a database of FIB fingerprints that come from known host-species. These techniques usually differ in the fingerprinting process, which the classification technique is dependent upon. Using these libraries, researchers can handle common FIB from a variety of host-species, making it incredibly agile. Disadvantages of this technique come from: the need to build a large library size which can become cost-prohibitive; the transient nature of some *E. coli* strains, assuming *E. coli* is the FIB of choice; and the fact that the applicability of the database is limited to the region in which the database was built from [60]. C-PLOP is a library-based MST technique *E. coli* as FIB and pyroprints as cost-effective fingerprints and researchers use it to understand *E. coli* strains and determine the host-species of fecal matter.

2.1 Pyroprinting

A key component of strain-based Microbial Source Tracking is the representation of strains of fecal indicator bacteria (FIB). Numerous genotypic methods exist for differentiation between strains of *E. coli*. One can find a detailed discussion of why pyroprints perform better than these options in [32].

Cal Poly researchers introduce the concept and construction of pyroprints in [9, 32], describing the process through which they construct pyroprints from the multiple loci of isolated *E. coli* DNA. It discusses the work done in [64], which confirmed the reproducibility of pyroprinting and determined that a Pearson correlation coefficient correlation above 0.99 "could be a good threshold to minimize false separation of isolates from the same strain." These works explain in detail the advantages of using the pyroprinting methodology with respect to cost ("[p]yroprinting could reduce the cost of a library-based MST investigation by up to 60%" [9]), reproducibility (much of which can be found in [64]), and discrimination between (known) strains of *E. coli*,

compared to existing state of the art methods. It asserts that while *E. coli* were used, the pyroprinting process applies to a broad range of bacteria whose genome contain multiple loci.

In-silico simulations done in [10] delved into the sensitivity of using the Pearson correlation coefficient ρ to compare constructed pyroprints of known *E. coli* alleles gathered from the National Center for Biotechnology Information database. CUDA programming on a GPU sped up the ρ computation considerably, allowing the researchers to understand, given all possible combinations of seven known alleles to form a simulated isolate, how many isolates are "*hard to differentiate*," i.e. have a $\rho_{ITS\text{-}1}$ and $\rho_{ITS\text{-}2}$ above 0.99 [10]. The work in [10] supplements *in-vitro* work performed in [42].

Senior projects and master's theses [56], [66], and [71] discuss the development of many of the tools in CPLOP, from the backend database construction, to the frontend web view and usage for investigation. Cal Poly researchers have placed a large emphasis on validation of the methodologies included in building pyroprints from *E. coli* isolates. Biology students investigated how *E. coli* strains change in response to a variety of factors. Computer science students at Cal Poly have developed many tools to aid the biologists in both validating their methodologies and performing *E. coli* strain research on various host-animals and host-species

2.2 Empirical Strain Research

CPLOP has enabled numerous research projects in the field of biology. The following is a list of empirical strain research performed using CPLOP:

- Using Hadoop to Identify False Positives in Bacterial Strain Typing from DNA Fingerprints [2]

- Demographics and Transfer of *E. coli* Within Bos taurus Populations [17]
- *E. coli* Strain Demographics and Transmission in Cattle [18]
- Application of Pyroprinting for Source Tracking of E. coli in Pennington Creek [44]
- Demographics of *E. coli* Strains in the Human Gut Using Pyroprints: A Novel MST Method [45]
- *Escherichia coli* Strain Diversity in Humans: Effects of Sampling Effort and Methodology [46]
- Investigating the Dominant *Escherichia coli* Strain in Lambs and Ewes Using Pyroprinting: A Novel Method for Strain Identification [48]
- Source Tracking of Fecal Contamination Along San Luis Obispo (SLO) Creek [62]
- Short Communication: Typing and Tracking Bacillaceae in Raw Milk and Milk Powder Using Pyroprinting [69]

These studies provide significant insight into the evolution and transmission of *E. coli* strains and demonstrate the effectiveness of using pyroprints and CPLOP as a MST method. Many of the above studies provided a culminating experience for undergraduates and graduates in biology and computer science. What we aim to provide with k-RAP is a set of tools that students and researchers have at their disposal to make it easier to make reproducible discoveries and assertions about strains in CPLOP.

2.3 Clustering

Presented in [42, 43] are the comparison of two hierarchical clustering techniques. *Primer5* [12] and a chronology-sensitive hierarchical clustering algorithm. Using metadata about when researchers collected the samples used to build the isolates, the hierarchical clustering proceeds to first cluster isolates from samples collected on the same day and continues to cluster by increasing days away from the initial collection date. They found that the clusters built by the chronology-sensitive hierarchical clustering algorithm resembled the *Primer5* clusters, but were unsure of whether these clusters were appropriate.

The work in [42, 43] went on to become a part of *OhClust!* (**O**ntology-Based **h**ierarchical **Clust**ering**!**) [67, 41, 40], a metadata-aware hierarchical clustering algorithm that allows CPLOP researchers to provide a metadata ontology to guide the order of hierarchical clustering. Hierarchical clustering in general is a very calculation-intensive process, making it a problematic tool for servers with limited computational power. The computational crux comes with the number of comparisons needed between clusters — clusters of isolates in CPLOP's case.

Most hierarchical clustering algorithms compute the distances between clusters and agglomerate by picking clusters to combine into a cluster (made of clusters) for the next hierarchy. Cluster distances are merely the distance between some representative member — possibly an average of the actual members — of one cluster with a representative from the other cluster. The representatives used to compute distance may be the members in each cluster that are, for example, closest to each other, farthest from each other, or the centroid of each cluster.

Computationally intense distance metrics make implementing a performant hierarchical clustering algorithm problematic for programmers. The way *OhClust!* gets

around this difficulty is by precomputing the distances — Pearson correlation coefficients in CPLOP — beforehand and storing them in memory. This greatly speeds up the clustering, but requires at least 4GB of memory for the distance lookup table alone. Since the servers that host CPLOP only have 4GB of RAM in total, *OhClust!* cannot be directly incorporated into CPLOP.

In [30], Eric Johnson presents a density-based clustering algorithm for pyroprints in order to build an intuitive clustering method that uses density and nearness and an efficient range query algorithm to find nearby isolates. DBSCAN [21], short for Density-Based Spatial Clustering of Applications with Noise, can be efficient if the distance metric used satisfies the triangle inequality. Unfortunately, Pearson correlation coefficient does not satisfy the triangle inequality, but work in [30] adjusts the comparison metric to use the euclidean distance of z-score normalizations and optimizes further by organizing the pyroprints into a tree, making DBSCAN a viable method of clustering for the servers that host CPLOP.

An attempt to use [30] as a naïve MST method in [37] revealed that for the isolates that actually clustered (i.e. were not determined to be noise), the accuracy was fairly high. Essentially, [37] clusters an unknown-host-species isolate along with the rest of the known-host-species isolates CPLOP and classifies it as the most plural host-species in the resulting cluster. However, [30] clustered only about half of the isolates in CPLOP, while the rest remained unclustered and thus unclassified. The investigation in [37] was useful to confirm suspicions of so-called "transient" strains of *E. coli* bacteria. Section 3.3.1 discusses the details of [30] relevant to this thesis, Section 4.1 describes a potential methodology to use it as a MST method, and Section 7.1 expands upon the investigation in [37] and determines whether it can be useful to supplement a MST method like k-RAP.

The clustering methods presented in [42, 43, 67, 41, 40] and [30] are examples

of typical investigations into bacterial strain research. On their own, they do not constitute an actual MST methodology[2]. Ultimately, the goal of CPLOP is to be able to objectively classify the host-species of an *E. coli* isolate. Thus, merely clustering isolates is insufficient for MST. This thesis presents k-RAP, an MST technique that works with the pyroprints of isolates in CPLOP as a solution to MST.

2.4 k-NN Techniques

A plethora of k-Nearest Neighbors (k-NN) methods exist, but most are various attempts to optimize the search space — either with efficient range queries or by leveraging information about the space to improve search speed — or modifications to the neighbor list structure to improve classification. Surveys on k-NN techniques [7, 29] show that each variation builds data structures for efficient query, or abstracts the notion of the usually euclidean distance metric to build a more accurate classifier, while others may weight the neighbors or remove neighbors from consideration based off of some criteria An exception to the typically euclidean distance metric comes in the way of recommender systems [15, 49], which use a notion of similarity based off of scores. While efficient range query interests us, we have a solution for it in [30].

The method that most closely resembles what we are after comes from techniques that build multiple k-NN classifiers by generating feature subsets and polling the classifier to determine the class of the unknown datapoint [4, 5, 70]. Similar to bagging and bootstrapping techniques used train other classification algorithms, the feature-set of known datapoints is either reasonably partitioned into feature-subsets [4, 5], or clustered into subsets [70]. Some even perturb the data and group features to create multiple k-NN classifiers [31]. The resulting classification from the k-NN subset is then aggregated and the final classification is determined by majority voting. This

[2]The work in [37] attempts to use clustering as a MST technique.

approach will not apply to CPLOP, since we do not merely have a single comparison metric that we want to partition into multiple to improve classification. Isolates in CPLOP always have two entirely separate metrics that we must make a reasonable decision from.

The primary goal of the k-NN Resolution Algorithms for Pyroprints (k-RAP) is to resolve the two comparison metrics that CPLOP has for comparing isolates. That is, given an isolate, in order to find nearby isolates, one must separately compute the Pearson correlation coefficient ρ for each ITS, giving us two comparison metrics, $\rho_{ITS\text{-}1}$ and $\rho_{ITS\text{-}2}$. Typically, the vectors in k-NN techniques represent the entire set of features for a particular datapoint. Many k-NN algorithms assume that the distance metric used — usually euclidean distance — will encode a useful notion of distance.

k-RAP can apply to other datasets with separate comparison metrics, especially those that contain types of features that euclidean distance does not apply to. For example, in a demographic study for, say, a political study, subjects may have a multitude of features with different metrics of comparison. Location of residence may be one and favorite color another[3], with a goal of classifying a subject's political party. Euclidean distance may not be appropriate for the location metric, since great-circle distance on a globe may encode closeness more accurately. For color, while it may be straightforward to represent red, green, and blue values as a vector, euclidean distance may not be the best choice to gauge similarity in color, certainly not in the same way as the great-circle distance, especially for the reasons put forth in [38], which discusses the tremendous difficulties in building a uniform perceptive color space. Simply combining these two features into a single vector and performing euclidean distance may not produce the most appropriate results. Nevertheless, these metrics on their own are perfectly amenable to their own, accordant distance metric

[3]Certainly, many other psychological metrics can exist, but for simplicity's sake, let us consider only these two.

that cannot necessarily be used on other features, making it easy to create a k-NN for each feature separately. As such, when using k-NN on datasets with a complex set of features there is a need for the ability to resolve separate k-NN lists in order to usefully classify datapoints.

Chapter 3

BACKGROUND

3.1 Biological

This chapter provides an overview of the problem that clustering for bacterial strain and k-Nearest Neighbors Resolution Algorithms for Pyroprints attempts to solve. Fecal contamination is a serious health concern that public officials work hard to mitigate effects of. Microbial Source Tracking is a solution typically that these officials typically employ in order to discover the source of fecal contamination and avert more contamination. Bacterial strain typing is just one of these techniques that The Cal Poly Library of Pyroprints employs in a cost-effective manner.

3.1.1 Fecal Contamination

Fecal contamination is dangerous for humans and animals alike. Pathogenic bacteria reside in fecal matter and contamination of publicly accessible and environmental water sources expose humans and animals to their dangerous effects. Often, it is up to natural resource managers and public health officials to both eliminate and prevent fecal contamination from occurring. Preventing contact with such bacteria is key to maintaining good public and environmental health.

Fecal matter can contain many different types of microbes, some of which are pathogenic. A *pathogen* is a microbe that causes disease in an animal. Pathogenic microbes that make their way into fecal matter include *Escherichia coli* (*E. coli*), *Salmonella typhii*, and *Campylobacter*. Diseases they might cause can include nose and ear infections, dysentary, and typhoid fever. Zoonotic pathogens — those that wild animals, pets, or livestock can transmit to humans — are of particular concern,

since it can be challenging to determine the source of the contamination. The use of antibiotics in livestock increases the resistance of some pathogens to traditional therapeutic techniques to treat the disease in humans [58]. It behooves public health officials and natural resource managers alike to mitigate the cause of fecal contamination for the safey and health of the public.

3.1.2 Microbial Source Tracking with Fecal Indicator Bacteria

Microbial Source Tracking (MST) aims to discover the host-species of microbial lifeforms, often employing Fecal Indicator Bacteria (FIB) to discover the source of fecal contamination. Many techniques exist that leverage unique characteristics of the FIB present in the fecal matter. Ultimately, choosing the right FIB requires that researchers and investigators understand their resource and budget constraints, and tailor their MST process accordingly.

MST techniques using FIB generally fall into two broad categories: library-dependent and library-independent. Certain FIB and bacteriophages are known to be from specific host-species, making host-species assessment of fecal matter possible without the need to build a library. However, most library-independent MST cannot establish a host-species source "beyond humans and a few domestic species" [58].

Library-dependent techniques build a library of known host-species samples of FIB and uses this data to determine host-species for an unknown sample. The effort needed to build such a library can be considerable, often requiring the genetic sequencing of a considerate number of samples to be effective [58]. Commonly, these libraries are only useful for the environment in which the researchers sampled from. What they gain by building such a library is the flexibility to determine the source of fecal matter from multiple host-species [58]. These library-dependent techniques rely on the of bacterial strain typing, discusses in the following section.

3.1.3 Delineating Bacterial Strains

When using FIB for MST, distinguishing between bacterial strains (BS) is a dynamic tool that can allow researchers to determine the source of fecal matter from a variety of host-species. Other methods that rely on MST are usually limited in the host-species they are effective at sourcing. By building a library of FIB samples, investigators can leverage bacterial strain typing to create a flexible and widely applicable method of sourcing fecal matter. Nevertheless, it has some drawbacks that, with some effort, MST investigators and researchers can overcome.

Bacterial strain typing for MST relies on the notion that strains of FIB remain mostly unique to the host-species from which they came. A *strain* of a species of microbe is a subtype of that species where the microbes in that strain are closely related to each other in some meaningful way. Generally, how one defines a strain differs between research groups, as each definition of a strain depends on the characterization of the microbe in question and the methods used to derive such characterizations. A strain, then, can be thought of as consisting of individuals descending from the same individual to generate a "group" or "family."

Escherichia coli (*E. coli*) inhabits the gut of many animals, but only certain strains are pathogenic. The guts of animals and humans are a great environment for flora like *E. coli* to thrive and multiply. *E. coli* can get into the gut in many ways based on the environment the host-animal lives in, the food they eat, and the other animals they interact with. Since *E. coli* resides in so many different species, researchers typically use it as the FIB of choice for library-based MST.

As [58] points out, the strains in *E. coli* can change considerably within the same host-species. Geography, time, rainfall, and habitat all can have an effect on the strains present in a particular host-species and even host-animal. The transient nature of some *E. coli* strains forces researchers to build a library that contains

a considerable number of samples to encapsulate the full spread of strains in the relevant host-species, adding to the cost and efforts needed to perform MST and usually limiting the environmental scope of the library built to the environment in which it was sampled from. The Cal Poly Library of Pyroprints attempts to remedy this by building a cheap yet reliable method of distinguishing between strains using pyroprints on the two *ITS* regions of *E. coli*.

3.2 The Cal Poly Library of Pyroprints

This section details the aspects of The Cal Poly Library of Pyroprints (CPLOP) relevant to microbial source tracking (MST). It explains the nature of pyroprints and the pyroprinting process, including what segments of *E. coli* DNA CPLOP researchers use and how they collect the isolates used in the process. Finally, it describes the steps necessary to properly compare two *E. coli* isolates for strain identification and MST and provides an overview of how CPLOP stores the pyroprint data to facilitate it.

3.2.1 Pyroprints

Pyroprints are the core data structure in CPLOP used to represent *E. coli* isolates. Using an inexpensive DNA sequencing technique called pyrosequencing, we can build a fingerprint[1] that allows us to effectively differentiate between *E. coli* strains. Building pyroprints requires careful use of the pyrosequencing process.

Pyrosequencing is a DNA sequencing technique appropriate for sequencing short DNA fragments (up to around 150-200 base pairs) [59]. A machine dispenses a predefined series of nucleotides and carefully measures the light output of the reaction. The amount of light emitted is directly proportional to the amount of the corresponding

[1] hence, the portmanteau pyroprint

nucleotide present in the DNA.

The output of the machine is a time series graph depicting the measured light before and after each dispensation. Previous work in [40, 64] helped us determine the optimal dispensation order and length and exactly which portions of this graph to use. Determining the optimal dispensation order and its length are crucial to effective sequencing: too few dispensations may not encode enough information, while too many may degrade the quality of the data.

A *pyroprint* is a vector representing the peak light values of the pyrosequencing of one of the *ITS* regions in the seven loci of the *E. coli* genome. As explained in Section 3.2.2, *ITS-1* and *ITS-2* offer keen insight into *E. coli* strains, since random variation can occur in them without affecting the survivability of the *E. coli* microbes. We pyroprint each *ITS* separately, building at least two pyroprints for each isolate: one for *ITS-1* and another for *ITS-2*.

3.2.2 Internal Transcribed Spacers

Choosing the proper region of DNA to fingerprint is crucial to effective strain delineation. Generally, when fingerprinting FIB, researchers avoid using regions that code for functional products and focus instead on non-coding regions of DNA, since variations within. When differentiating between *E. coli* strains, researchers use the two *ITS* regions between three genes: *16S*, *23S*, *5S*.

The *16S*, *23S*, and *5S* ribosomal RNA operons (rDNA) are genes that help code for proteins in *E. coli* bacteria. Any changes to these regions may affect the rate or nature of protein synthesis and thus affect the survivability of the bacteria. As a result, we consider these regions to remain *conserved* across *E. coli* strains. Using these segments directly to differentiate between strains would be fruitless, since even wildly different *E. coli* strains will still have nearly identical copies of these three

regions.

Between these three genes are two non-coding, and thus *unconserved*, regions, called internal transcribed spacer (*ITS*). Since *ITS* do not code for functional products, random variations occur in *ITS* regions that do not affect the survivability or reproducibility of the bacteria. Researchers frequently use *ITS* for strain delineation, due to their unconserved nature. Importantly, any offspring of a microbe inherit the *ITS-1* and *ITS-2* regions, allowing biologists to use them to differentiate strains [67]. The two *ITS* regions that bridge *16S-23S* and *23S-5S* we respectively refer to as *ITS-1* and *ITS-2*. Amplifying the *ITS* regions of DNA becomes a straightforward and inexpensive process, due to the highly conserved regions flanking each *ITS*. Primers can reliably attach to the rDNA immediately next to each *ITS*, because of their conserved nature, allowing for polymerase chain reaction (PCR) amplification of the *ITS-1* and *ITS-2* regions.

Applying PCR to an *E. coli* isolate requires awareness of the following principle, crucially affecting how we can interpret a fingerprint:

Principle 3.2.1 (Repeated Loci)**.** The *ITS* regions of *E. coli* and the rDNA — referred to collectively as *loci* — repeat around the *E. coli* genome seven times.

Figure 3.1 depicts these seven loci and the relative position of the *ITS* regions between the rDNA. The primers used to attach to the rDNA attach to each of the seven instances, resulting in PCR amplification of all seven copies of *ITS-1* and *ITS-2*. What results from PCR is an amplified mixture of these seven unconserved *ITS* regions that we can use as a fingerprint for the isolate.

The repeating of the *ITS* regions makes pyroprints different from traditional pyrosequencing. Traditional pyrosequencing allows researchers to figure out the sequence of nucleotides that make up the segment of DNA, because they only pyrose-

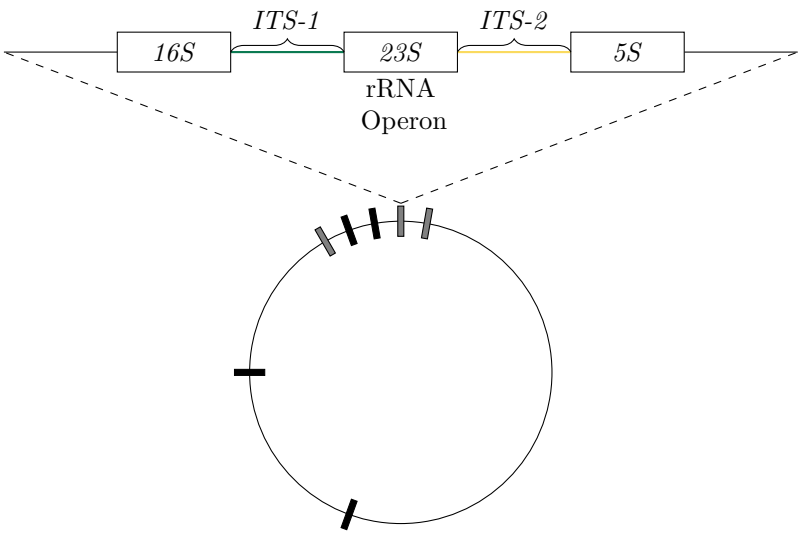


Figure 3.1: A diagram of a simplified segment of *E. coli* DNA, outlining the *ITS-1* and *ITS-2 ITS* regions, which repeat 7 times around the *E. coli* genome.

quence the PCR amplification of a single segment — or multiple conserved segments — at a time. Pyroprinting considers the PCR amplification of more than one unconserved segment of DNA at a time, encoding more information with a single pyroprint. As a result, CPLOP researchers cannot use a pyroprint to figure out the nucleotide sequence of the pyroprinted *ITS* region.

3.2.3 Obtaining & Comparing *Escherichia coli* Isolates

Cal Poly students collect *Escherichia coli* (*E. coli*) isolates from the fecals samples of a variety of different sources and compare them for a multitude of different studies. Culturing and *E. coli* extraction occurs in an introductory cell and molecular biology class. Comparing two isolates requires separate consideration of each *ITS* region.

The data stored in CPLOP are obtained as follows. Biologists collect fecal sample from a host-subject of a known species. They extract *E. coli* and culture individual bacterial cells from the bacterial material contained in each sample. A bacterial *isolate* is an individual culture grown from a fecal sample. Each isolate undergoes

PCR procedures that amplify the DNA in the two *ITS* regions of DNA, after which the pyrosequencing of each region produces pyroprints that are stored in the CPLOP database.

Sources of isolates in CPLOP are often animal species, but include many isolates cultured from environmental sources, like creeks and the ocean. A large portion of CPLOP consists of isolates derived from Cow and Human sources. The disproportionate number of Cows in CPLOP is due to a study investigating the strain demographics and transmission in cattle [18, 17]. Every year, Cal Poly houses cattle from around the state starting in May for testing and vaccination before they leave for auction in September. Cal Poly researchers obtained fecal sample from every cow as they arrived and when they departed, comparing the isolates for similarity before and after cohabitation. Isolates derived from Humans make up a large proportion of CPLOP because Cal Poly students investigated *E. coli* strain characteristics in a variety of studies [46, 43, 45]. Such disproportionate representation of host-species in library-based MST is a common problem and Chapter 7 discusses the effect with respect to CPLOP.

The nature of separately pyroprinting the *ITS-1* and *ITS-2* regions of an isolate requires adherence to the following principle:

> **Principle 3.2.2** (Comparing Isolates)**.** The only valid comparison between isolates using the a comparison metric ρ is a separate comparison of the *ITS-1* pyroprints using ρ and *ITS-2* pyroprints using ρ: Given two isolates Isolate A and Isolate B, where Isolate A has pyroprints $\textit{ITS-1}_A$ and $\textit{ITS-2}_A$ and Isolate B has pyroprints $\textit{ITS-1}_B$ and $\textit{ITS-2}_B$, $\rho(\textit{ITS-1}_A, \textit{ITS-1}_B)$ and $\rho(\textit{ITS-2}_A, \textit{ITS-2}_B)$ are the only valid comparisons between the two isolates.

In other words, in order to compare two isolates, one must consider the pyroprints of each *ITS* region separately, due to the fundamental concept of strains and why we

pyroprint these two regions. Figure 3.2 depicts the comparison process. Comparing an *ITS-1* pyroprint to an *ITS-2* pyroprint with ρ is completely meaningless, since the two pyroprints represent different segments of DNA in the *E. coli*.

3.2.4 Pearson Correlation Coefficient

Strain delineation underlies the goals of CPLOP — the ability to encode the concept of strains of the FIB *E. coli* and distinguish between different ones is fundamental to library-based MST. Towards that end, CPLOP researchers need some way to compare isolates using the pyroprints of their *ITS* regions that effectively distinguishes between different strains. Picking the right comparison function to compare the pyroprints of two isolates is crucial and we found that the Pearson Correlation Coefficient is an effective metric.

Given two D-dimensional vectors, $\vec{x} = (x_1, \ldots, x_D)$ and $\vec{y} = (y_1, \ldots, y_D)$, the Pearson correlation coefficient ρ is:

$$\rho(\vec{x}, \vec{y}) = \frac{1}{D} \sum_{i=1}^{D} \frac{(x_i - \mu_{\vec{x}})(y_i - \mu_{\vec{y}})}{\sigma_{\vec{x}} \cdot \sigma_{\vec{y}}} = \frac{cov(\vec{x}, \vec{y})}{\sigma_{\vec{x}} \cdot \sigma_{\vec{y}}} \tag{3.1}$$

where $\mu_{\vec{x}}$, $\mu_{\vec{y}}$ are the means of x_i's and y_i's respectively, $\sigma_{\vec{x}}$, $\sigma_{\vec{y}}$ are their standard deviations, and $cov(\vec{x}, \vec{y})$ is the covariance between the two vectors. The Pearson correlation coefficient encodes a notion of similarity; as the final portion of Equation 3.1 shows, ρ calculates the covariance of the two vectors, normalizing the value by the standard deviation of both.

The *covariance* of two vectors measures the joint variability of the vectors, i.e. how much one vector varies with respect to another. Given two D-dimensional vectors,

$\vec{x} = (x_1, \ldots, x_D)$ and $\vec{y} = (y_1, \ldots, y_D)$:

$$cov(\vec{x}, \vec{y}) = \frac{1}{D} \sum_{i=1}^{D} (x_i - \mu_{\vec{x}})(y_i - \mu_{\vec{y}}) \tag{3.2}$$

Positive covariance between two vectors means the two behave similarly, while negative means they behave in the opposite manner.

The *standard deviation* measures the amount of deviation a set of values has from its average. Given a D-dimensional vector $\vec{x} = (x_1, \ldots, x_D)$, its standard deviation is:

$$\sigma_{\vec{x}} = \sqrt{\frac{1}{D} \sum_{i=1}^{D} (x_i - \mu_{\vec{x}})^2} \tag{3.3}$$

One can also define the standard deviation as the square root of the covariance:

$$\sigma_{\vec{x}} = \sqrt{cov(\vec{x}, \vec{x})} \tag{3.4}$$

Using the Pearson correlation coefficient as a measure of similarity is straightforward, due to it being a combination of the covariance and the standard deviation. Since one can define the covariance of a vector with itself in terms of the standard deviation:

$$cov(\vec{x}, \vec{x}) = \sigma_{\vec{x}}^2 \tag{3.5}$$

it is clear that given two vectors, $\vec{x}$ and $\vec{x}\,'$, where $\vec{x} = \vec{x}\,'$:

$$\rho(\vec{x}, \vec{x}\,') = \frac{cov(\vec{x}, \vec{x}\,')}{\sigma_{\vec{x}} \sigma_{\vec{x}\,'}} = \frac{cov(\vec{x}, \vec{x})}{\sigma_{\vec{x}} \sigma_{\vec{x}}} = \frac{\sigma_{\vec{x}}^2}{\sigma_{\vec{x}}^2} = 1 \tag{3.6}$$

Due to the normalizing effect of the σ's, it is impossible for $\rho > 1$. Conversely, two very dissimilar vectors will have a $\rho < 1$. Specifically, Pearson correlation coefficient measures a linear correlation between vectors, so two vectors with a $\rho = 0$ means

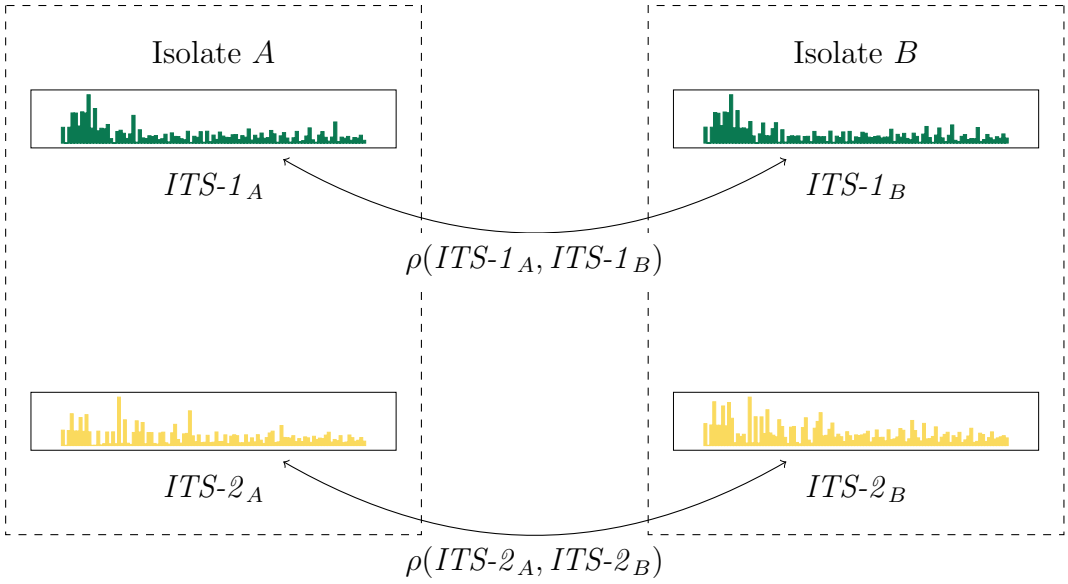


Figure 3.2: Comparing isolates involves comparing the pyroprints of each isolate using ρ, the Pearson correlation coefficient, with the stipulation that one can only compare pyroprints from the same *ITS* in their respective isolate. The green bar plots represent a pyroprint of *ITS-1*, while gold bar plots represent a pyroprint of *ITS-2*.

that the two vectors have no linear relationship[2]. Work done in [64] determined that multiple pyroprints of the same isolate obtain a $\rho > 0.995$ and CPLOP researchers use this value for quality control [32, 9].

It is from this measure of similarity ρ that we define a comparison metric between pyroprints. The nature of this function works perfectly for what pyroprints encode. The values in a pyroprint represent peak light intensity values from chemical reactions, the intensity of which is proportional to the nucleotide content of the DNA pyroprinted. Peak intensity differences and noise from the machine are accounted for by Pearson correlation coefficient, since it measures how the pyroprint vector values change with respect to each other and machine variations will be similar between pyroprintings.

[2]For completeness, if $\rho \approx -1$, then there is an *inverse correlation* between the two vectors. It is easy to see if given $\vec{x} = (x_1, \ldots, x_D)$ and $\vec{x}\,' = (x'_1, \ldots, x'_D) = (-x_1, \ldots, -x_D)$, then $cov(\vec{x}, \vec{x}\,') = -cov(\vec{x}, \vec{x}) \Rightarrow \rho(\vec{x}, \vec{x}\,') = -\rho(\vec{x}, \vec{x}) = -1$. By similar reasoning, $-1 \leq \rho \leq 1$

Defining strains using the Pearson correlation coefficient requires a similarity threshold, above which we may consider two isolates to be part of the same strain. Two isolates are considered to be of the same strain if the pyroprints of both regions have a $> \alpha$, where $\alpha = 0.990$ [64, 9]. Work done in [64] determined this α threshold by simulating the pyroprinting process with tools from [40] on known *E. coli* strains from the National Center for Biotechnology Information database. Simulations performed in [10] further confirmed the usefulness of this α value.

3.2.5 Database

There are three pieces that comprise CPLOP: the physical cold storage of fecal samples and isolates, the backend data store, and the frontend web interface. Cold storage allows CPLOP researchers to perform The backend data store holds the pyroprints and metadata about their host-species and collection. The web interface, shown in Figure 3.3, allows researchers to perform queries and test whether isolates match.

Cold storage holds the collected fecal samples and the isolates cultured from them. It allows CPLOP researchers to culture additional isolates and re-pyroprint existing isolates. Often, researchers refer to the cold storage as the "library" in library-based MST.

The data store for CPLOP pyroprints is a MySQL database. It stores metadata for each collected sample, including who collected it and where and when they collected it. Importantly, the name of the host-species and a unique designation for the host-animal marks the sample that the pyroprints of an isolate came from. For computationally intense

The web frontend, written in PHP, allows CPLOP to access the information in the database from the Internet to: perform queries for isolates and pyroprints, browse

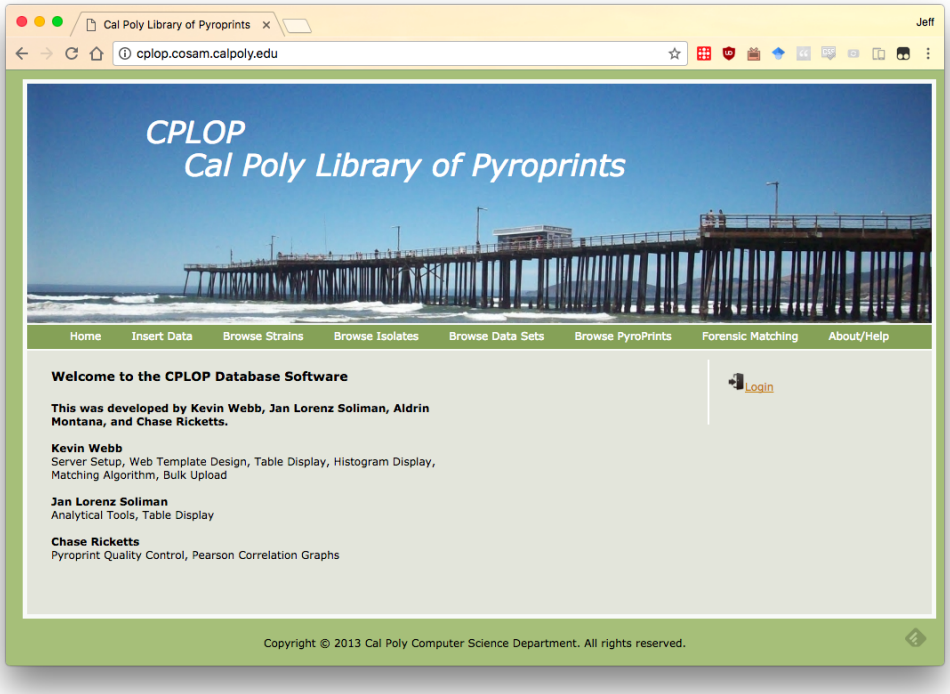


Figure 3.3: Researchers use CPLOP through a web-accessible frontend.

isolate and pyroprint datasets, and perform forensic matching. The isolates in CPLOP are visible from this interface (see Figure 3.4) as are the pyroprints. Isolates may have multiple pyroprints, which the CPLOP frontend allows researchers to browse, which Figure 3.5 shows. Certain isolates come from particular collection runs, be they certain studies or classroom examples, and appear collectively as datasets on the website. CPLOP also provides the ability to view the pyroprint histogram, as Figure 3.7 shows. Forensic matching (Figure 3.8) is a crucial feature of CPLOP, allowing researchers to query a dataset against the CPLOP database to find matching isolates.

Cal Poly servers host the CPLOP website at `http://cplop.cosam.calpoly.edu/`. The servers are limited in computational ability, only containing 4GB of RAM. Such limitations make it difficult to implement algorithms like *OhClust!* [40, 41, 43], which require more than 4GB of RAM for efficient computation, or [2], which requires

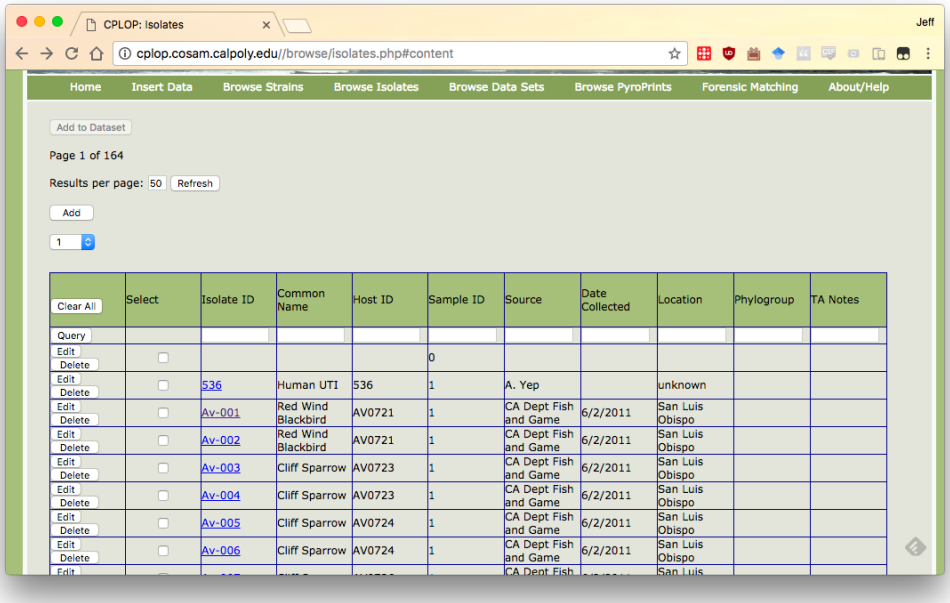


Figure 3.4: CPLOP allows researchers to explore its isolates

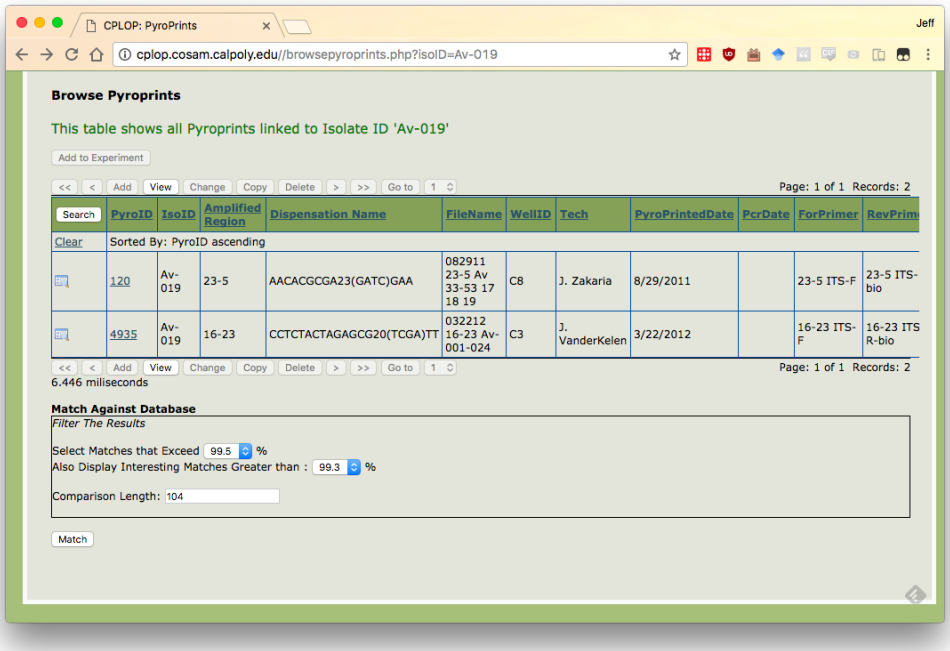


Figure 3.5: The CPLOP frontend allows researchers to browse the pyroprints of an isolate.

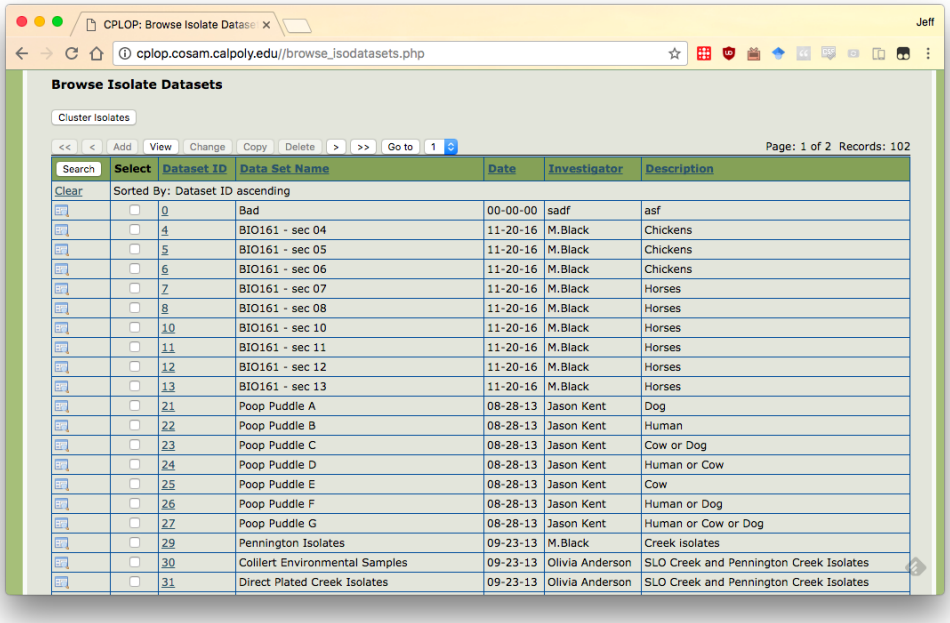


Figure 3.6: Certain isolates may be part of a collection of isolates, which CPLOP has the ability to sort by.

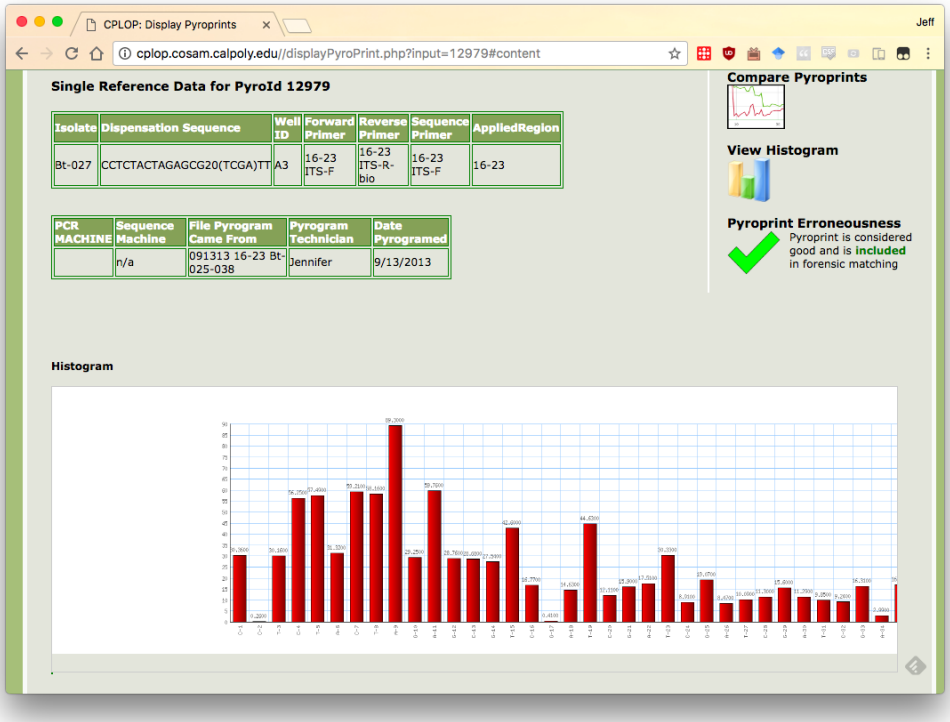
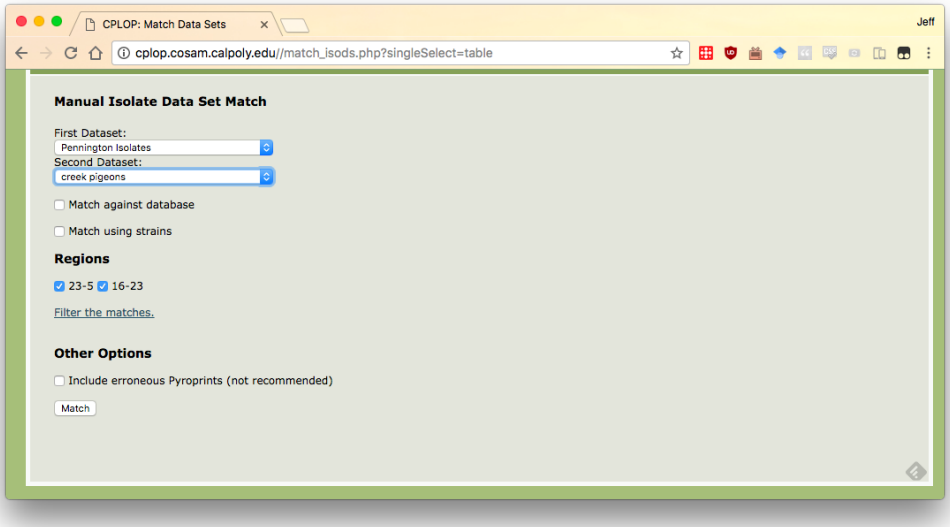
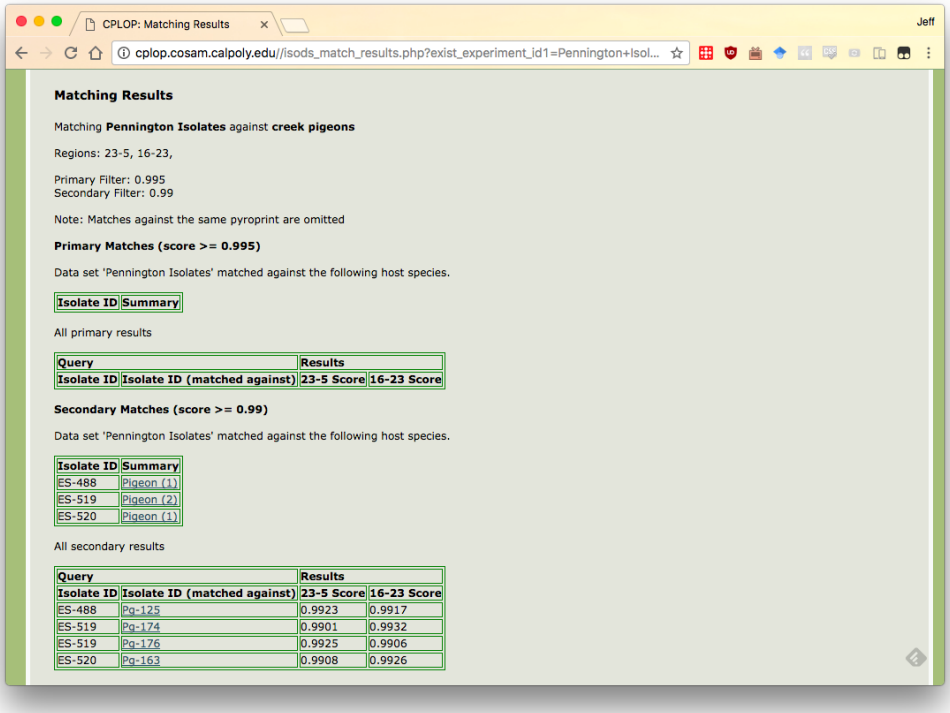


Figure 3.7: Researchers can view the histogram of an individual pyroprint using CPLOP.



(a)



(b)

Figure 3.8: Forensic matching is a key feature of CPLOP, allowing researchers to choose subsets of CPLOP data (a) to find strain-level matches (b).

access to a cluster of computers for *MapReduce* ability. Future work will assess the feasibility of moving over to more dynamic systems, like Amazon Web Services.

3.3 Computational

Two computer science concepts are core to the clustering and classification techniques investigated in this thesis. One is a clustering algorithm that builds clusters by categorizing the datapoints into three different types, clustering some and denoting the unclustered as noise. Another is a classification technique that searches for nearby datapoints in order to classify an unknown datapoint.

3.3.1 Density-Based Clustering of Isolates

Density-based clustering algorithms build clusters based on two parameters: the minimum number of neighbors `MinPts`, a point must have to be a core point of a cluster, and ε, the radius that those neighbors must be within. These algorithms define clusters with respect to core points and border points — points within ε of a core point — labeling everything else — the singletons — as noise. For this work, we chose to use DBSCAN as the clustering technique for grouping isolates because dense groupings of similar isolates fits our intuition of bacterial isolate strains. Closely related "families" of isolates will appear in the same cluster and we want these clusters to have sufficient purity to aid us in MST.

DBSCAN[21] provides the framework for our clustering algorithm. It uses a distance metric, a minimum neighbors value `MinPts`, and an ε range to categorize data points as one of three types: a core point, a border point, or noise. A *core point* is a point that has at least `MinPts` data points within ε of it. A *border point* is a point that is within ε of a core point, but that does not have `MinPts` points within ε of it. Every other point is *noise*. A *cluster* is a group of neighboring core points with their

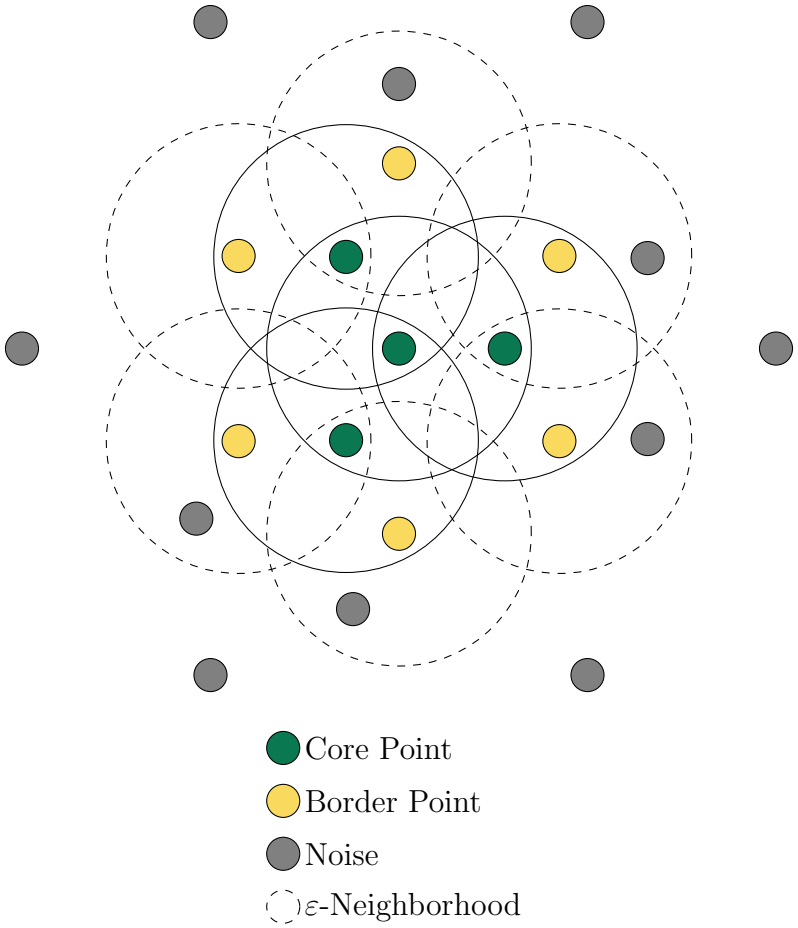


Figure 3.9: A basic density-based clustering with `MinPts` = 3 points and a unit ε represented by the circles — solid for the core neighborhoods and dashed for border.[4] We see that the green points each contain 3 neighbors, but while the border points do not, they are within ε of a core point and we thus cluster it along with the core points. The (single) cluster that results from this set of datapoints are the green and gold points depicted.

associated border points. According to this definition of a cluster, all clusters must have at least `MinPts` points in them. Figure 3.9 depicts this process.

Density-based clustering techniques require a distance metric, often times the euclidean distance, between data points in order to cluster. Performing fast range queries greatly improves the speed of clustering. If the range query can finish in $O(\log n)$ time, then DBSCAN can run in $O(n \log n)$ time. Organizing the data into a spatial index can optimize these spatial queries.

Spatial indexes structure the data into a search tree, similar to a binary search tree, organizing the points by distance. When querying for nearby points, the algorithm can traverse this search tree, ignoring certain points along the way. While this can speed up the range query to a $O(\log n)$, many spatial indexes degenerate into a $O(n)$ operation. In the former case, this makes DBSCAN run in $O(n \log n)$ time.

In DBSCAN, the `RangeQuery` function handles range queries by taking as parameters the data point and a distance and returning all data points within range of the query point. Mathematically, a *range query* is a function $RangeQuery : D \times \mathbb{R} \rightarrow \{D\}$ that takes a query point $q \in D$ and a real-valued $\varepsilon \in \mathbb{R}$ and returns $\{d \in D | Dist(q, d)\} < \varepsilon\}$ — the set of all other points within ε of the query point — where $Dist$ is some distance metric $Dist : D \times D \rightarrow \mathbb{R}$. If the distance metric forms a proper metric space, then data structures like quad trees and octrees can speedup ε range queries for a point. One can imagine the range query as a hypersphere centered at the query point with a radius of the query range. In order to make `RangeQuery` fast, we had to make some optimizations.

3.3.2 k-Nearest Neighbors

The k-Nearest Neighbors classification algorithm (k-NN) is a straightforward algorithm to classify an unclassified object using a library. Using a comparison function, it compares the unclassified object to "nearby" classified objects. It uses the concept of a comparison function to formulate an idea of "closeness," asserting that the similarity an unknown object has to a class of objects relates to the class of the unknown objects itself. To outline the process: Given an unclassified object u, a library of classified objects $\mathbb{L}$, and a comparison function, $\mathbb{C}$:

1. Compare u to each object in $\mathbb{L}$ using $\mathbb{C}$
2. Add the classified object and the result to a list of neighbors, N

3. Sort N by most similar

4. Consider only the top k entries in N, called the k-nearest neighbors

5. Classify u as the *most plural* classification in the k-nearest neighbors list

Algorithm 1 describes this process in pseudocode.

Algorithm 1 k-Nearest Neighbors

Input:

- $u \in \mathbb{U}$: an unknown isolate
- $\mathbb{L} \subseteq \mathcal{I}$: a library of known isolates

Output:

- $\mathcal{S}$ value, classifying the unknown isolate u

Requires:

- k, α, and the comparison function C are predetermined

```
1: procedure ClassifyKNN(u, 𝕃)
2:     N ← ∅                          /* Make nearest neighbors list.        */
3:     for p ∈ 𝕃 do                    /* For each library element:           */
4:         sim ← C_i(u, p)                    /* Compare.                    */
5:         add (sim, p) to N                  /* Add to neighbors.           */
6:     end for
7:     sort N by sim                  /* Sort by most similar.               */
8:     s ← FindMostPluralSpecies({n_1, ..., n_k | n_i ∈ N})
9:     return s
10: end procedure
```

The motivation is that the unclassified object must be "close" to some of the classified objects in our database, using an appropriate measure of closeness — the *comparison function* — for the data. By choosing the *most plural* classification — the classification that shows up the highest number of times — in the k-nearest neighbors we can, with some accuracy, classify our unknown object.

Chapter 4

METHODOLOGY

There are two main components we investigated for the purpose of MST: clustering for bacterial strains and the k-Nearest Neighbors Resolution Algorithms for Pyroprints (k-RAP). The clustering method we use is based off of the density-based clustering algorithm DBSCAN, as introduced in Section 3.3.1 and described in [30]. The k-RAP derives its classification ability from k-Nearest Neighbors, outlined in Section 3.3.2, adding four methods to resolve the multiple k-nearest neighbors lists that are a product of multiple neighbor-comparison functions and an α threshold to filter the k-nearest neighbors lists. This Chapter describes the use of these two methods as classification methodologies for CPLOP.

4.1 Clustering for Bacterial Strains

In order to understand the make up of bacterial strains in CPLOP, we chose to investigate how a density-based clustering algorithm might group the isolates we have collected so far and how that might affect a rudimentary MST technique based off of clustering: cluster an unknown isolate along with the isolates in CPLOP and classify it as the most plural species of the cluster. Density-based clustering ties in well with our notion of closely-related strains of *E. coli* (the relation being the separate Pearson correlation coefficient comparison of each *ITS* region we use to compare isolates). From the computer science point of view, a bacterial strain is essentially a cluster of *E. coli* isolate representations stored in CPLOP. Our MST method, thus, works as follows:

1. **Strain Identification.** Identify bacterial strains in CPLOP by clustering all

CPLOP isolates.

2. **MST.** Given an isolate of unknown origin, find the cluster it belongs to. Return the host-species of the plurality of isolates in the cluster.

Our clustering algorithm is the density-based clustering algorithm developed by Johnson [30]. It extends DBSCAN for the case of two comparison functions between data points (our isolates are compared based on the two *ITS* regions) and implements an efficient spatial data structure to manage the retrieval of the data points.

DBSCAN can easily use an efficient range query technique to find nearby points and speed up clustering time considerably by taking advantage of the triangle inequality that proper metric spaces have. Unfortunately, Pearson correlation coefficient does not encode a metric space, because it fails the triangle inequality[1]. This complicates range queries, discussed in Section 3.3.1, because spatial indexes tend to rely on the triangle inequality, usually with euclidean distance, to argue that certain points can be ignored during a spatial index tree traversal.

To allow us the use of a spatial data structure with the Pearson correlation coefficient to store data points during the clustering procedure we use instead the euclidean distance pyroprint z-score normalizations, which we derive by recognizing in Equation 3.1 that Pearson correlation coefficient is made up of z-score normalizations of $\vec{x}$ and $\vec{y}$. The z-score normalization of $\vec{x}$ is:

$$z(x_i) = \frac{x_i - \mu_{\vec{x}}}{\sigma_{\vec{x}}}$$

where $\mu_{\vec{x}}$ and $\sigma_{\vec{x}}$ are the mean and standard deviation of the values in a single pyroprint respectively. Thus, for clustering, we compare pyroprints using the Euclidean

[1] $d(x, z) \leq d(x, y) + d(y, z)$

distance d of z-scores.

$$d(\vec{z}_{\vec{x}}, \vec{z}_{\vec{y}}) = \sqrt{\sum_{i=1}^{D}(z(x_i) - z(y_i))^2} \tag{4.1}$$

where D is the number of dimensions. This allows us to use spatial indexes and $O(\log n)$ lookup in DBSCAN.

Each isolate is represented in CPLOP by a pair of pyroprints: one each from each *ITS-1* and *ITS-2* region, complicating the use of DBSCAN and the meaning of α threshold. We handle this in DBSCAN by performing two range queries, one each for *ITS-1* and *ITS-2*, taking the intersection of the two results. We must, however, pick a suitable ε for each *ITS* region.

CPLOP uses a threshold value of $\alpha = 0.995$ to compare two pyroprints. Pyroprints with Pearson correlation coefficient above α are considered to represent the same DNA material, while pyroprints with a Pearson correlation coefficient below α are considered to represent different DNA material [64, 66, 67].

The number of dispensations D used to build a pyroprint differ for the *ITS-1* and *ITS-2* regions. Because $D_{ITS\text{-}1} \neq D_{ITS\text{-}2}$, the original α under the space defined by (4.1) no longer applies in the same way to both regions. An alternative formulation of (4.1), with respect to the Pearson correlation coefficient ρ, is:

$$d(\vec{z}_{\vec{x}}, \vec{z}_{\vec{y}}) = \sqrt{2 \cdot D \cdot \rho(\vec{x}, \vec{y})} \tag{4.2}$$

where D is the number of dimensions and $D_{\vec{x}} = D_{\vec{y}} = D$. Using Equation 4.2, we can convert α to the values in Table 4.1. We use these converted α values as the ε for each *ITS* region's `RangeQuery`.

When clustering CPLOP isolates using our density-based clustering algorithm,

Table 4.1: Converted α threshold to fit the new metric space defined by (4.1).

***ITS* Region**	*ITS-1* α	*ITS-2* α
$\rho(\vec{x}, \vec{y})$	0.995	0.995
D	96	94
$d(\vec{z}_{\vec{x}}, \vec{z}_{\vec{y}})$	0.9747	0.9644

we need to set up the two parameters at our disposal: `MinPts` and ε. For ε we choose the two values shown in Table 4.1 converted from the 0.995 Pearson correlation coefficient threshold of pyroprint similarity. Essentially, we only want to consider the ε-neighborhood of a pyroprint that contains the other pyroprints that we consider to represent the same DNA material.

For the `MinPts` parameter, we use *grid search* running our clustering with `MinPts` set to $1, 2, 3, 4, 5, 6,$ and 7. The `MinPts` value adjusts how strict our definition of a cluster is. That is, the higher the value of `MinPts`, the more neighbors a core point must have with ε of it and its neighbors to become a cluster. Balancing this value with the coverage of our algorithm is crucial to its success, because for too low of a value, we may not have a clear plurality in a cluster, while too high of a value may miss some smaller clusters that might classify our unknown isolate into something other than noise.

4.2 k-RAP

Due to the two-*ITS*-region nature of *E. coli* isolates in CPLOP[2], we effectively have two comparison functions between datapoints (isolates), complicating our use of k-NN. k-NN provides CPLOP biologists a transparent and intuitive way of understanding the host-species classification it asserts, so we find that it will be a usefully insightful . Applying k-NN to CPLOP isolates gives us two lists, one for each *ITS*

[2]see Section 3.2.3

region, that we must make a classification from. In order to accommodate multiple comparison functions, we need a multiple-k-nearest neighbors list resolution strategy. Rather than create a new similarity metric out of a pair of similarity scores, which may deviate from the inherent nature of the comparison function, we choose to update the k-NN method with four different ways of selecting the resultant category label: the k-Nearest Neighbors Resolution Algorithms for Pyroprints (k-RAP). These four methods are described below.

In what follows, we generalize our problem. Given u and v, two library objects (isolates), and a collection of comparison functions, $\mathbb{C}= (\mathbb{C}_1, \ldots \mathbb{C}_m)$, with $m > 1$, comparing u to v gives us a collection of values: $\mathbb{C}(u, v) = (\mathbb{C}_1(u, v), \ldots, \mathbb{C}_m(u, v))$. All four resolution procedures described in this section work with such a generalized representation of isolates and comparison functions between them.

Given an unknown isolate u, a library of classified[3] isolates $\mathbb{L}$, and a set of comparison functions $\mathbb{C}$, we compare u to each object in $\mathbb{L}$ using each comparison function in $\mathbb{C}$. To resolve these comparison functions, we propose four algorithms:

4.2.1 Comparing Isolates

Comparing isolates to each other is of primary interest to biologists using CPLOP. CPLOP represents each isolate by a pair of mutually incomparable pyroprints: one for each of the two *ITS* regions. As a result, given isolates I_1, I_2, we can represent each as a pair of pyroprint vectors

$$I_1 = (\vec{q}_1, \vec{q}_2) \text{ and } I_2 = (\vec{r}_1, \vec{r}_2),$$

where $\vec{q}_1$ and $\vec{r}_1$ are respectively I_1 and I_2's *ITS-1* pyroprint and $\vec{q}_2$ and $\vec{r}_2$ are respectively I_1 and I_2's *ITS-2* pyroprint [9]. Since pyroprints from different regions are

[3] A "classified isolate" is an isolate for which the host-species has been identified in the database.

incomparable, comparing isolates must be done as follows:

$$\mathbb{C}(I_1, I_2) = (\rho(\vec{q_1}, \vec{r_1}), \rho(\vec{q_2}, \vec{r_2})),$$

where $\rho(\cdot, \cdot)$ is between pyroprints of the same *ITS* region and is the Pearson correlation coefficient. Thus, when comparing isolates, we effectively have two different similarity metrics, one for each *ITS* region:

$$\mathbb{C}(I_1, I_2) = (\mathbb{C}_1(I_1, I_2), \mathbb{C}_2(I_1, I_2)).$$

Algorithm 2 Isolate Comparison Metric

Input:

- $u \in \mathcal{I}$: an isolate
- $v \in \mathcal{I}$: an isolate

Output:

- $\mathbb{R}$ value, indicating similarity

Requires:

- Each isolate has a pyroprint in the i^{th} ITS region

1: **procedure** $C_i(u, v)$
2: $\quad \vec{p_u} \leftarrow \textsc{GetPyroprint}_i(u)$
3: $\quad \vec{p_v} \leftarrow \textsc{GetPyroprint}_i(v)$
4: $\quad$ **return** $\textsc{PearsonCorrelation}(\vec{p_u}, \vec{p_v})$
5: **end procedure**

4.2.2 α Filtering

Our first modification to k-NN is an additional condition at step 4, after finding the k-nearest neighbors:

4. Consider only the top k entries in N above threshold α

The α threshold allows biologists to filter out neighbors that are among the k closest, but too dissimilar to compare. When comparing multiple pyroprints of the same region of a single isolate for quality control, the Pearson correlation coefficient between them is strictly above 0.995. As a result, for many other studies — not necessarily MST-focused — CPLOP researchers use a Pearson correlation coefficient of 0.990 or above to define a strain of *E. coli*. Filtering by some value near this may give more accurate results and provides an intuitive way to relate these lists to other studies.

Algorithm 3 k-NN with α Threshold

Input:

- *list*: sorted list

Output:

- *nearest*: the top k elements above the threshold α

Requires:

- k and α are predetermined
- each element in *list* has a similarity field *sim*

```
1: procedure FILTER_{k,α}(list)
2:     nearest ← ∅
3:     i ← 0
4:     while i < k and list[i].sim < α do
5:         add list[i] to nearest
6:         i ← i + 1
7:     end while
8:     return nearest
9: end procedure
```

4.2.3 Meanwise Resolution

For u and a $p \in \mathbb{L}$, we take the mean of the result of all of the comparison functions and build a single k-nearest neighbors list from it. The mean can be any metric mapping $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and in the investigated implementation, we use the euclidean distance, also known as the L^2 norm. A single k-nearest neighbors list

results from this algorithm that we filter by k and α and use to classify the unknown. Algorithm 4 describes this process in pseudocode.

Algorithm 4 Meanwise Resolution

Input:

- $u \in \mathbb{U}$: an unknown isolate
- $\mathbb{L} \subseteq \mathcal{I}$: a library of known isolates

Output:

- $\mathcal{S}$ value, classifying the unknown isolate u

Requires:

- k, α, and the set of comparison metrics $\mathbb{C}$ are predetermined

```
1: procedure ClassifyMean(u, 𝕃)
2:     N ← ∅                          /* Make nearest neighbors list.          */
3:     for p ∈ 𝕃 do                   /* For each library element:             */
4:         𝔸 ← {∅}                    /* Make empty set for results.           */
5:         for C_i ∈ ℂ do             /* For each comparison metric:           */
6:             sim ← C_i(u, p)                          /* Compare.          */
7:             add sim to 𝔸                             /* Track result.     */
8:         end for
9:         mean ←Mean(𝔸)              /* Mean the results.                     */
10:        add (mean, p) to N         /* Add mean to neighbors.                */
11:    end for
12:    sort N by sim                  /* Sort by most similar.                 */
13:    N ←Filter_{k,α}(N)             /* Keep the nearest.                     */
14:    s ← FindMostPluralSpecies(M)
15:    return s
16: end procedure
```

4.2.4 Resolution by Winner

For each comparison function, we make a k-nearest neighbors list and filter by k and α accordingly. Once we finish building each comparison function's k-nearest neighbors list, we find the most plural classification from each list and track the number of times that classification shows up in that list. Then, we classify u based off the classification that has the highest number in its corresponding list. Algorithm 5

describes this process in pseudocode.

Algorithm 5 Resolution by Winner

Input:

- $u \in \mathbb{U}$: an unknown isolate
- $\mathbb{L} \subseteq \mathcal{I}$: a library of known isolates

Output:

- $\mathcal{S}$ value, classifying the unknown isolate u

Requires:

- k, α, and the set of comparison metrics $\mathbb{C}$ are predetermined

```
1: procedure ClassifyWinner(u, 𝕃)
2:     N ← ∅                          /* New list to track neighbor lists.        */
3:     for C_i ∈ ℂ do                 /* For each comparison metric:              */
4:         N_i ← ∅                        /* Make nearest neighbors list.         */
5:         for p ∈ 𝕃 do                   /* For each library element:            */
6:             sim ← C_i(u, p)                /* Compare.                         */
7:             add (sim, p) to N_i            /* Add to neighbors.                */
8:         end for
9:         sort N_i by sim                /* Sort by most similar.                */
10:        N_i ←Filter_{k,α}(N_i)         /* Keep the nearest.                    */
11:        add N_i to N
12:    end for
13:    S ← {∅}                        /* To track each list's most plural.        */
14:    for N_i ∈ N do
15:        s ← FindMostPluralSpecies(N_i)
16:        add s to S
17:    end for
18:    return Max(S)                  /* The most plural overall.                 */
19: end procedure
```

4.2.5 Resolution by Union

For each comparison function, we make a k-nearest neighbors list and filter by k and α accordingly. After building each k-nearest neighbors list, we combine the lists into a set, keeping track of the original list position for tie-breaking. From this set, which we dub the union, we count the classifications present in the union and classify

u as the most plural in the union of the lists, compared to the other lists. Algorithm 6 describes this process in pseudocode.

Algorithm 6 Resolution by Union

Input:

- $u \in \mathbb{U}$: an unknown isolate
- $\mathbb{L} \subseteq \mathcal{I}$: a library of known isolates

Output:

- $\mathcal{S}$ value, classifying the unknown isolate u

Requires:

- k, α, and the set of comparison metrics $\mathbb{C}$ are predetermined

```
1: procedure ClassifySetwise(u, L)
2:     M ← {∅}                          /* Create new common set.          */
3:     for C_i ∈ C do                   /* For each comparison metric:     */
4:         N_i ← ∅                      /* Make nearest neighbors list.    */
5:         for p ∈ L do                 /* For each library element:       */
6:             sim ← C_i(u, p)          /* Compare.                        */
7:             add (sim, p) to N_i      /* Add to neighbors.               */
8:         end for
9:         sort N_i by sim              /* Sort by most similar.           */
10:        N_i ←Filter_{k,α}(N_i)       /* Keep the nearest.               */
11:        for n ∈ N_i do               /* For each nearest neighbor:      */
12:            add n to M               /* Add to common set.              */
13:        end for
14:    end for
15:    s ← FindMostPluralSpecies(M)
16:    return s
17: end procedure
```

4.2.6 Resolution by Intersection

For each comparison function, we make a k-nearest neighbors list and filter by k and α accordingly, but ensure that we do not lose track of the entire sorted list of results. After building each k-nearest neighbors list, we inspect each list for common isolates. We add isolates that appear in every list into a set that we call the inter-

section. If the size of the intersection is k, then we are done. Otherwise, we increase the length of our individual lists by δ and search for common isolate. This process repeats until the size of the intersection is k, or all of the isolates in the individual lists are below threshold α. Algorithm 7 describes this function in pseudocode.

Algorithm 7 Resolution by Intersection

Input:

- $u \in \mathbb{U}$: an unknown isolate
- $\mathbb{L} \subseteq \mathcal{I}$: a library of known isolates

Output:

- $\mathcal{S}$ value, classifying the unknown isolate u

Requires:

- k, α, δ, and the set of comparison metrics $\mathbb{C}$ are predetermined

```
1: procedure ClassifyIntersection(u, 𝕃)
2:     N ← ∅                          /* New list to track neighbor lists.        */
3:     for C_i ∈ ℂ do                 /* For each comparison metric:              */
4:         N_i ← ∅                        /* Make nearest neighbors list.         */
5:         for p ∈ 𝕃 do                   /* For each library element:            */
6:             sim ← C_i(u, p)                /* Compare.                         */
7:             add (sim, p) to N_i            /* Add to neighbors.                */
8:         end for
9:         sort N_i by sim                /* Sort by most similar.                */
10:        add N_i to N
11:    end for
12:    done ← false
13:    while ¬done do
14:        for N_i ∈ N do
15:            N'_i ← ∅
16:            N'_i ←Filter_{k,α}(N_i)
17:        end for
18:        if |N'_1 ∩ ··· ∩ N'_n| < k then
19:            k ← k + δ
20:        else
21:            N_∩ ← N'_1 ∩ ··· ∩ N'_n
22:            done ← true
23:        end if
24:    end while
25:    return FindMostPluralSpecies(N_∩)
26: end procedure
```

4.2.7 CPLOP Makeup

Figure 4.1 shows the distribution of CPLOP isolates considered in this study among its 53 different host-species.

There are a total of 4,610 isolates in our dataset[4]. As seen from Figure 4.1, the organic growth of CPLOP yielded disproportionately many *E. coli* isolates originating from humans and cows (however, as shall be seen below, these isolates belong to a large number of strains). Each isolate is represented in CPLOP with two pyroprints — one each for *ITS-1* and *ITS-2* region.

[4]A simplified version of CPLOP containing isolate IDs, host-species, and z-score normalizations can be found at `https://github.com/jmcgover/cplop-acm-bcb-2016`.

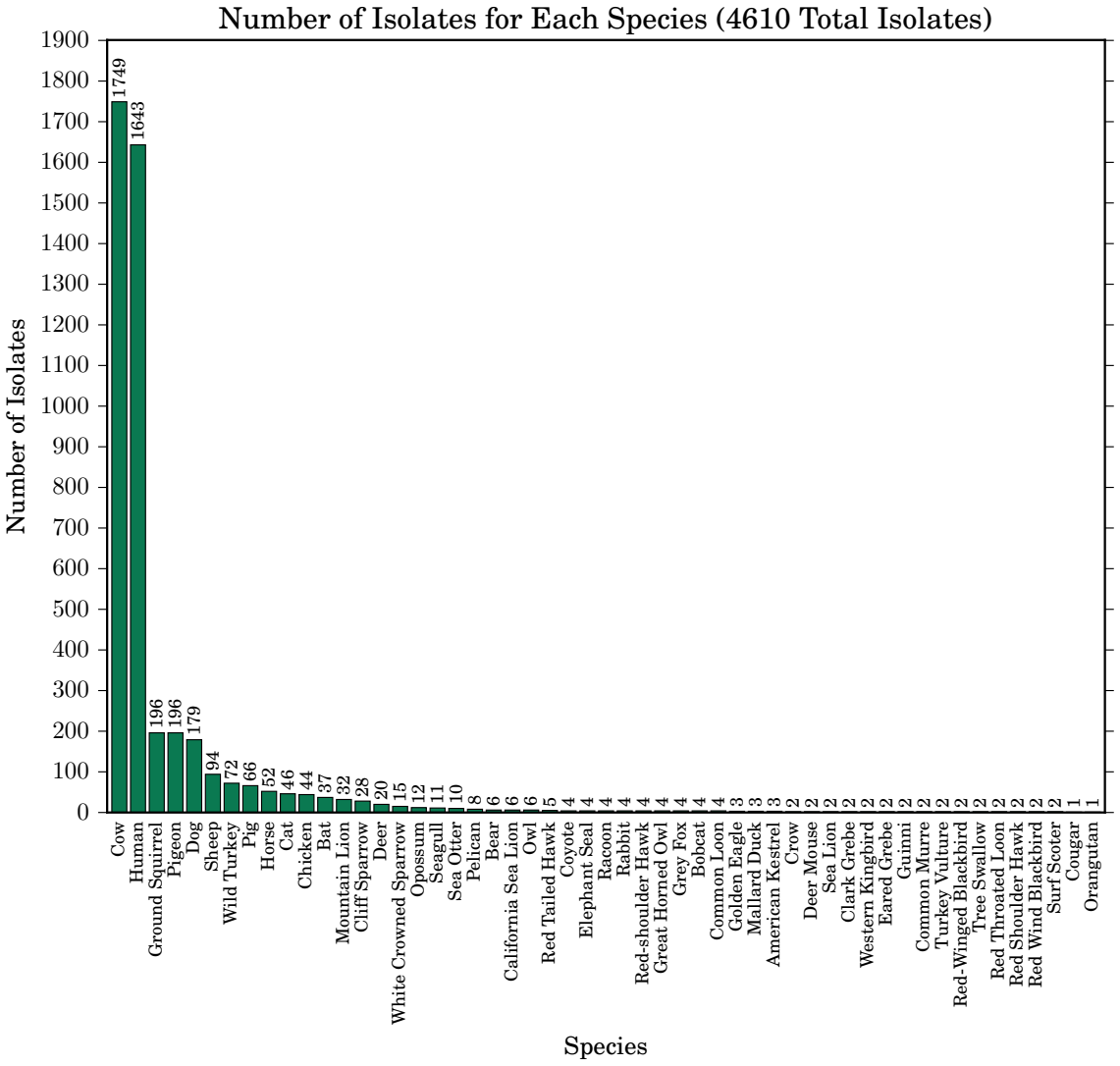


Figure 4.1: A histogram of the number of isolates of each species in our study, taken from CPLOP. There are 4,610 total isolates from 53 different host-species.

Chapter 5

IMPLEMENTATION

5.1 Graphing Cluster Metrics

Forthcoming...

5.2 Resolution Algorithms

Forthcoming...

Chapter 6

EVALUATION

Evaluating the bacterial strains derived from density-based clustering and classification accuracy of k-RAP is key to gauging the effectiveness of these two techniques. For the density-based clustering method, we focused on cluster purity and clustering coverage. For k-RAP, we investigate precision and recall and use an indicator that balances both of them called the F_1-measure.

6.1 Cluster Purity and Clustering Coverage

Clustering for bacterial strains has two aspects that we must evaluate: how pure the bacterial strains (clusters) are and how many of the isolates in CPLOP end up in a cluster (as opposed to noise). The former tests whether the *E. coli* strains stay relatively unique to the host-species from which they come from, the core theory of library-based MST. The latter tells us how effective Section 4.1 describes how we can use clustering as a MST method — cluster an unknown isolate along with the isolates in CPLOP and classify it as the most plural species of the cluster. Cluster purity can easily gauge how effective this technique is at MST, since the concept readily summarizes to how pure, from a host-species perspective, a cluster is. Density-based clustering algorithms such as DBSCAN may not cluster every datapoint, as mentioned in Section 3.3.1, labeling some datapoints (isolates) as noise and thus leaving them unclustered.

In this paper we look at the results of clustering CPLOP data using this algorithm from the perspective of cluster purity. We call a cluster (bacterial strain) *100% pure* if all isolates that belong to it come from the same host-species.

Of interest to us is the following information:

1. The number of 100% pure clusters and the percentage of bacterial isolates from CPLOP clustered into pure clusters.

2. The structure of impure clusters: specifically, whether a dominant host-species can be clearly identified in each cluster.

3. Coverage: the total number of CPLOP isolates found to belong to a strain.

4. MST Accuracy: the percentage of isolates for which the strain-based MST procedure produces the correct response.

Thus, our core measure is *cluster purity*, the proportion of a cluster that comes from the most plural host-species of that particular cluster. A *100% pure cluster* is a cluster which only contains data points (isolates) with the same class label (same host-species of origin).

Consider a cluster $C = \{c_1, \ldots, c_K\}$. Let $s(c)$ refer to the species of isolate c. Let m be the plurality species label for data points in C, and let the total number of points in C with $s(c) = m$ be s_m. Then the *individual cluster purity* ν of cluster C is:

$$\nu(C) = \frac{s_m}{K}$$

In addition to computing the purity of individual clusters we want to have an understanding of the overall purity on the entire dataset. Given a *clustering* $\mathcal{C} = \{C_1, \ldots, C_n\}$ on a dataset, we define the size $\mathcal{M}$ of the set of clusters:

$$\mathcal{M} = \sum_{i=0}^{n} |C_i| \tag{6.1}$$

The *overall clustering purity* is:

$$\sum_{i=1}^{n} \frac{|C_i|}{\mathcal{M}} \cdot \nu(C_i) \tag{6.2}$$

One can think of (6.2) as a form of weighted arithmetic mean of the purities, where the size of the cluster adds more weight to the value.

Coverage of the dataset is important to an effective MST method. The density-based clustering method we use has one key disadvantage: a clustering run with the parameter `MinPts`, treats all points that do not fit into a cluster of size of at least `MinPts` as noise. This means that as the value of `MinPts` grows, so will the number of isolates that do not cluster into a strain.

Given the parameter `MinPts` of the clustering algorithm, we collect the following four measures, that collectively represent the breakdown of all data points (isolates) in CPLOP:

1. *Noise.* Number/percentage of isolates clustered as noise points.
2. *Misses.* Number/percentage of isolates from minority species in impure clusters.
3. *Hits.* Number/percentage of isolates from plurality species in impure clusters.
4. *Pure points.* Number/percentage of isolates in 100% pure clusters.

6.2 Host-Species Classification

Evaluating k-RAP requires an understanding of how well it classifies the host-species of an isolate. There are a few areas of focus that we have when interpreting the results of k-RAP:

- What size k achieves the best results?

- What size α achieves the best results?
- Which metric resolution algorithm achieves the best results?

Indeed we can define "best" in many ways, but we choose to look at two metrics, recall and precision, and a combination of the two, the F-measure. The metrics look at the accuracy of the classification on the object and the object on the classification respectively, while F_1-measure hopes to represent a balance between the two. We test k-RAP by performing cross validation with holdout.

6.2.1 Cross Validation with Holdout

To gauge the effectiveness of k-RAP at classifying the host-species of an isolate, we cross-validated against the library by separately holding out each isolate in CPLOP from CPLOP, classifying it against CPLOP, and verifying whether it is correct. Since each isolate in CPLOP has the correct host-species, we know whether a classification is correct or not.

6.2.2 Recall

In our study, recall tracks how well we are able to discover all isolates from a given category, i.e. with a given host species. Given a category (host-species name), the recall for that host species is the percentage of isolates taken from this host species that have been properly identified. For example, if our database had 100 cat isolates, and 74 of them were classified by our method as having come from a cat, the recall would be 74%. In this study, we compute both overall recall (what percentage of isolates were classified as their proper host-species label) as well as host-species-level recall (what percentage of isolates that came from dogs/humans/sheep/etc. were classified as their proper label).

6.2.3 Precision

Precision tracks how well our method avoids misclassification errors. Given a category and a list of isolates our method classified as belonging to it, the precision of the method on the category is the percent of isolates from the list that has the correct label. For example, if our method returned 100 isolates labelled "Dog" of which 77 isolates really did come from dogs, the precision of the method is 77%. As with recall, we compute both overall precision, as well as the precision for each category/species label.

6.2.4 *F*-Measure

The F_1-measure, F_1, is the *harmonic mean* of the precision, P and the recall, R:

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = 2 \cdot \frac{P \cdot R}{P + R}$$

While we prefer maximizing this value, a value near 0.5 means we are doing well.

Chapter 7

RESULTS

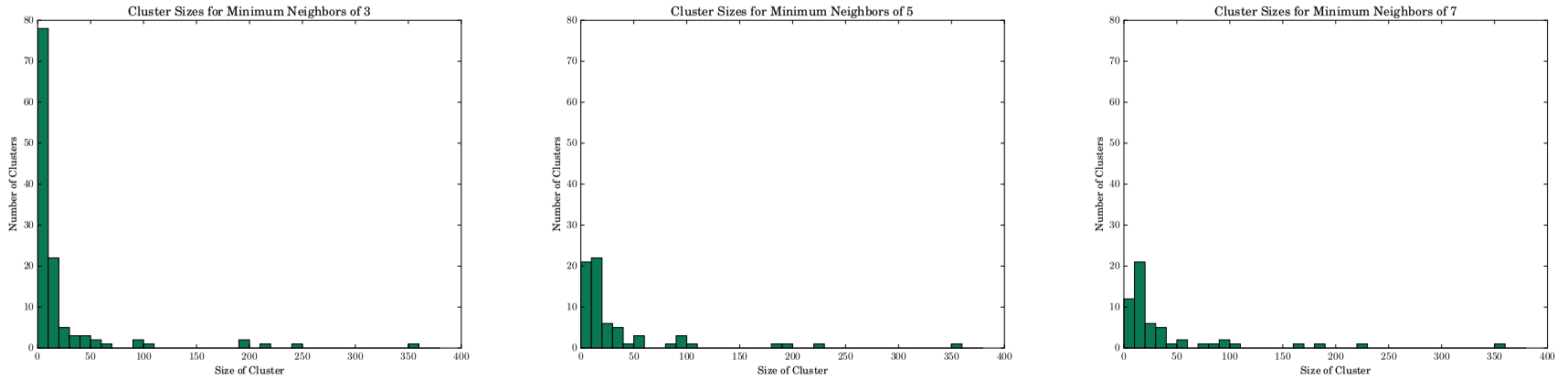
Clustering for bacterial strain and investigating the k-RAP led us to focus on a volley of metrics in order to understand the nature of the data. For clustering, we were interested in both the nature of the clusters — the size, purity, and unique host-species within each — as well as the clustering coverage — how many CPLOP isolates made it into a cluster, given a specific `MinPts` value. For k-RAP, we adjusted the values of k and α for each resolution strategy to gauge how effective each were at classification.

7.1 Clustering

In gauging how effective our clustering method is against CPLOP, we looked at the distribution of cluster sizes, the number of isolates that fell into high purity clusters, the number of unique species in each cluster and how that affected the size and purity, and overall coverage and accuracy metrics. From these data, we gained some insight into the clustering algorithm and were able to visualize some predictions we had about the biological aspects of strains.

7.1.1 Cluster Size Distribution

Figure 7.1 shows the distribution of cluster sizes as `MinPts` increase from 3, to 5, to 7. We see that at all three `MinPts` values, the number of small clusters (fewer than 10) dominates the overall makeup of clusters. Figure 7.1 shows a propensity towards small clusters at low `MinPts` values. This creates a high number of 100% or almost 100% pure clusters. Most clusters are tiny, with a few larger clusters for small `MinPts`



(a) Cluster Size Distribution for `MinPts` of 3 (b) Cluster Size Distribution for `MinPts` of 5 (c) Cluster Size Distribution for `MinPts` of 7

Figure 7.1: The size distribution of clusters skews heavily towards smaller clusters.

values.

As we approach higher `MinPts` values, the smaller clusters disappear. As `MinPts` increases from 3 to 5, we lose over half of clusters of size smaller than 10; while as `MinPts` increases to 7, we lose only a few more. Furthermore, while the number of clusters with 10-20 isolates stays relatively stable across `MinPts` values, the number of clusters with 50-100 isolates increase for `MinPts` of 5 and 7.

7.1.2 Cluster Purity Distribution

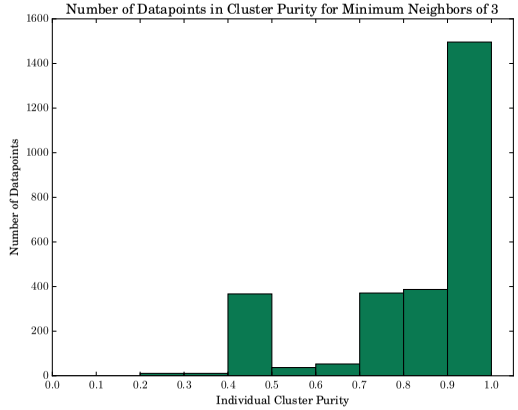
Of interest to our investigation is the number of isolates that fall within clusters of a particular purity. Figure 7.2 shows the number of isolates that fall within a cluster of a particular purity as a histogram. We notice that as `MinPts` increases, the purity skews towards purer clusters and that a portion of isolates remain in an impure cluster regardless of the `MinPts` value.

From `MinPts` of 3 all the way to 7, there are about 400 isolates that land in a cluster of purity between 0.4 and 0.5. This group of isolates remains largely unchanged as we restrict the `MinPts` value. We suspect (and discuss in Section 7.1.6) that certain *E. coli* strains find themselves in many host-species fecal matter.

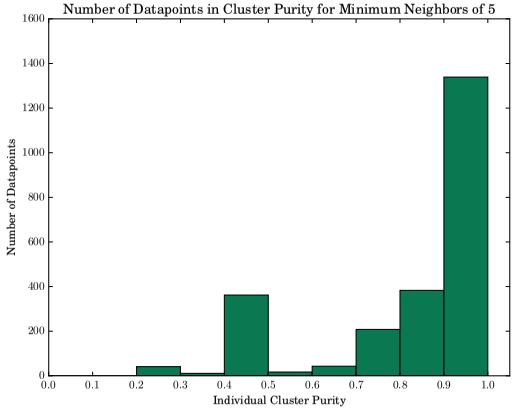
7.1.3 Unique Species in Each Cluster

Knowing the number of unique host-species in our clusters is key to understanding how our strain-based MST algorithm performs. Figure 7.3 plots the number of unique species in each cluster (vertical axis) against individual cluster purity (horizontal axis) representing each cluster as a circle of diameter proportional to cluster size[1]. The points at the lower right represent many clusters of various size of 100% (or near so)

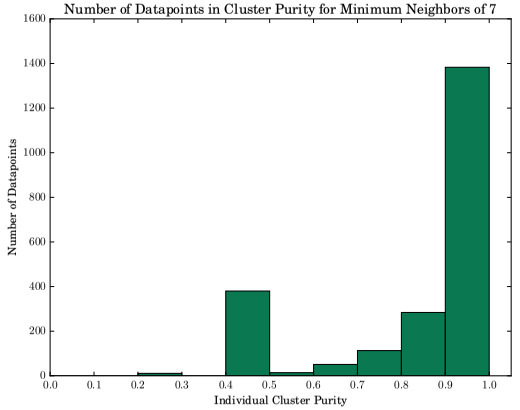
[1]A linear scaling of the diameter with respect to *the largest cluster amongst all the clusterings* defines the diameters of the dots.



(a) Cluster Purity Distribution for `MinPts` of 3

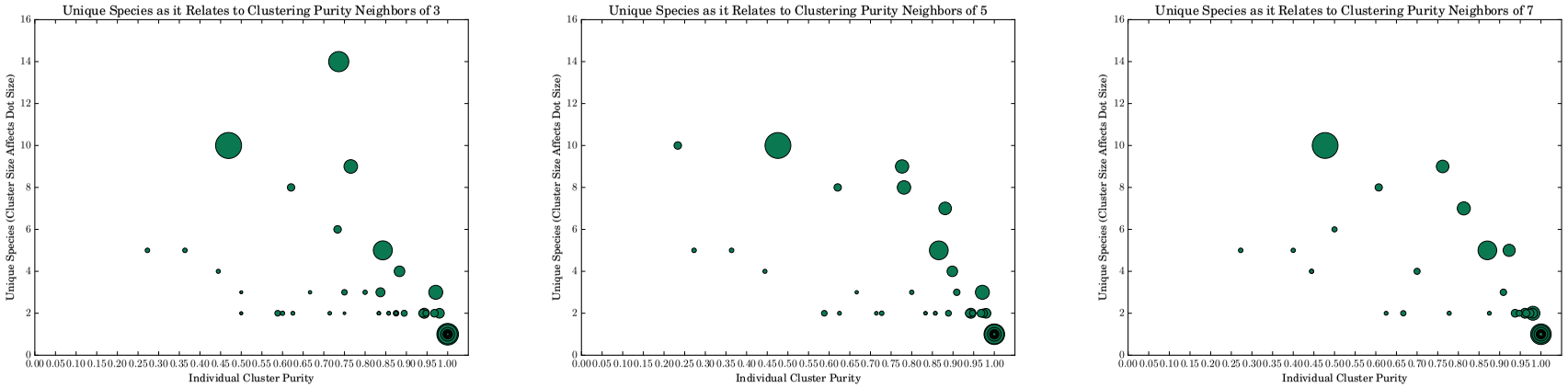


(b) Cluster Purity Distribution for `MinPts` of 5



(c) Cluster Purity Distribution for `MinPts` of 7

Figure 7.2: The number of isolates that fall into a cluster of a given purity. We notice that that the number of isolates that fall into the 0.90 to 1.00 cluster purity range decreases as we increase `MinPts` from 3 to 5, but increases from 5 to 7.



(a) Cluster Purity for `MinPts` of 3

(b) Cluster Purity for `MinPts` of 5

(c) Cluster Purity for `MinPts` of 7

Figure 7.3: These three dimensional graphs show the individual cluster purity in the horizontal axis, the number of unique species in the vertical axis, and the relative size of the cluster in the diameter of the dots. Each individual dot is its own cluster. We find that as we restrict cluster to needing more neighbors (increasing the `MinPts` value), we lose some clusters and gain more pure clusters.

purity and are stacked from largest behind cluster to smallest in front.

As `MinPts` values increase we see one cluster at the top right (`MinPts`=3) with 14 unique species disappear as `MinPts` becomes 5. One very low purity cluster at a `MinPts` value of 5 disappears when we increase `MinPts` to 7 in Figure 7.3c.

A particularly large cluster at around 0.45 purity with 11 unique host-species, remains relatively intact (and is clearly recognizable) as `MinPts` changes from 3, to 5, to 7. This can account for the large amount of isolates clustered into impure clusters in Figure 7.2.

As we restrict the cluster size with `MinPts`, we see that this appears to break up some clusters and cause others to become bigger. It is difficult to track exactly how a cluster changes without making some simplifying assumptions or without tracking all 4,610 isolates as they move from cluster to cluster.

7.1.4 Clustering Coverage

Clustering coverage is important to consider, since we want our clustering algorithm to apply to as many isolates as possible. Towards this end we investigated the four metrics introduced in Equation 6.1 — noise, misses, hits, and pure points — for each `MinPts` clustering investigated. We hope to find the `MinPts` value that gives us the most pure points, but will also settle for the fewest misses, shown in Figure 7.4.

The cyan area is noise — isolates that were not clustered. The dark green area is the proportion of pure points. Light green is the number of hits. Gold is the number of misses.

It is good to note that the number of misses are low and flatten out as we increase `MinPts` from a value of 3, giving us good reason not to investigate clustering where `MinPts` is greater than we have already investigated. The number of pure points

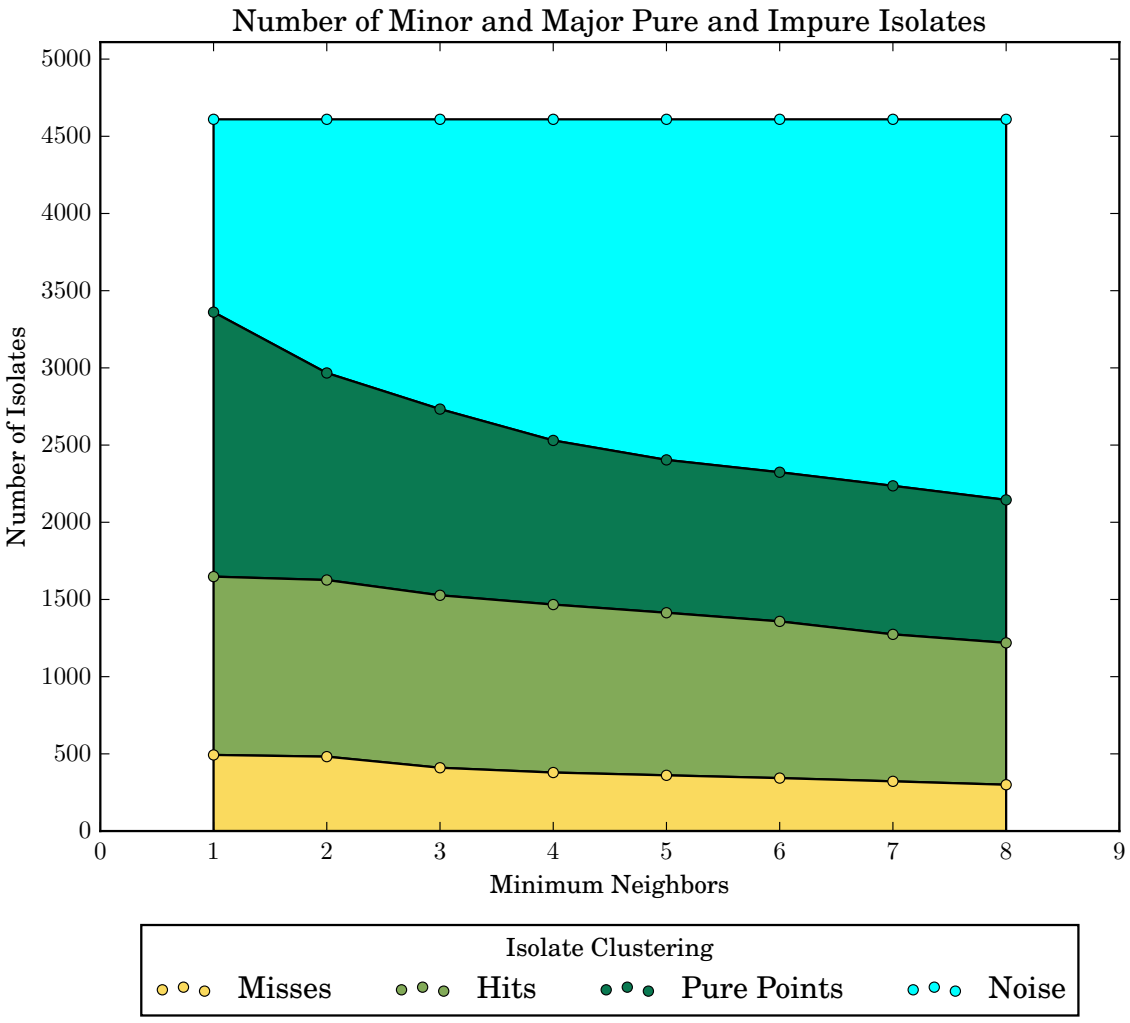


Figure 7.4: As `MinPts` increases, we see that we cluster fewer isolates. throughout, the number of major pure isolates stays relatively equal to the number of major impure isolates.

stays relatively equal to the number of hits. Important in Figure 7.4 is the amount of isolates that the algorithm does cluster. The combination of the gold and two green areas show the total number clustered, while the cyan shows the number of isolates that were *not* clustered. It is unfortunate that the number of noise isolates is high, but we plan to mitigate that in future work.

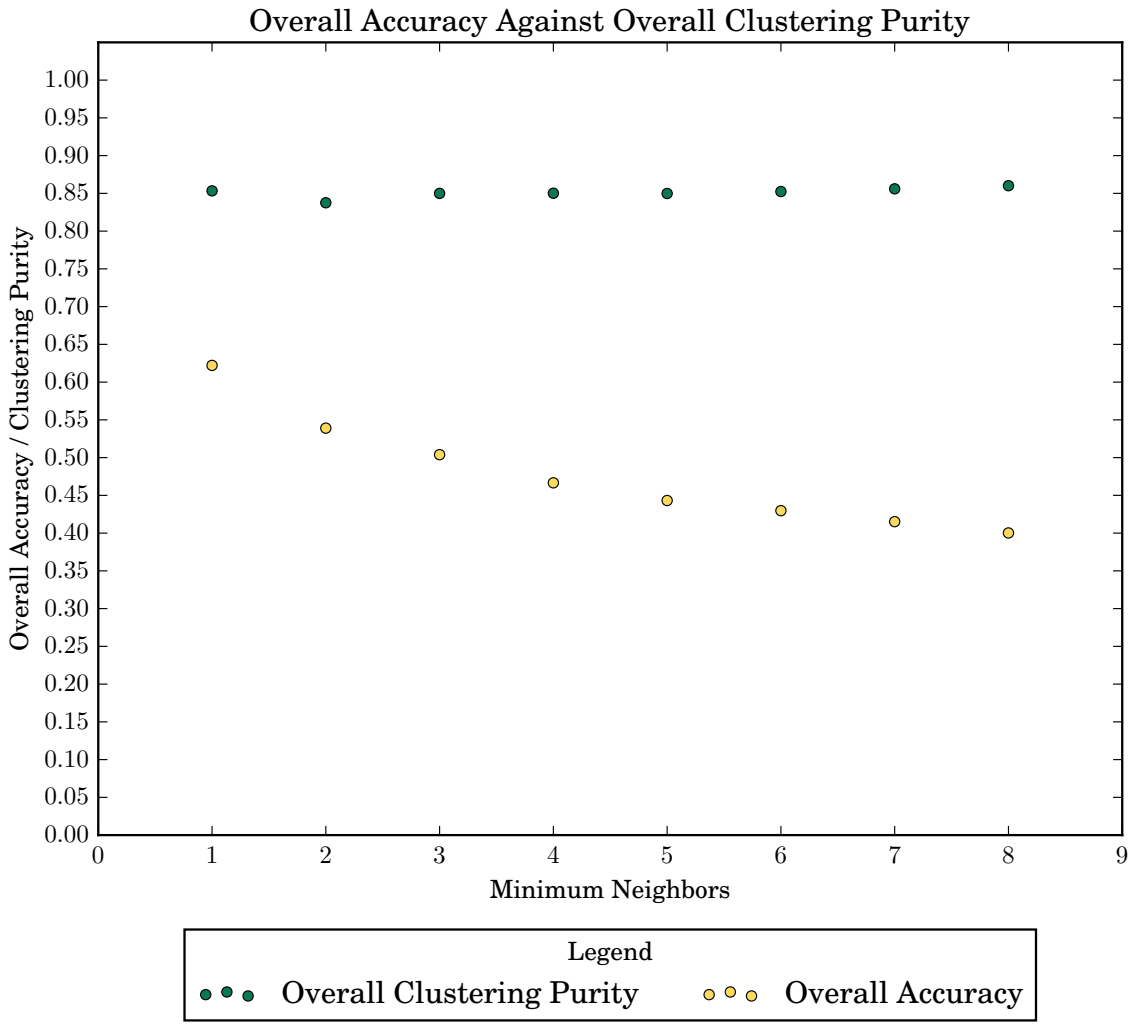


Figure 7.5: The overall accuracy decreases as we restrict `MinPts`. The overall clustering purity stays relatively the same as we increase the value for `MinPts`. That is, for clustered isolates, the classification algorithm stays relatively the same relative to the number of isolates accurately clustered.

7.1.5 Overall Clustering Purity

Overall clustering purity, defined in (6.2), is the number of isolates clustered that end up in a cluster where their host-species is the most plural host-species. The overall accuracy is the proportion of correctly classified isolates out of all the isolates under consideration. We want to maximize both values, but would prefer the former over the latter. Coverage is an issue we are concerned about, but we plan to mitigate

this issue by leveraging [36] against clusters of isolates.

Figure 7.5 shows the overall accuracy compared to the overall clustering purity. A `MinPts` value equal to 3 is the last `MinPts` value where the overall accuracy stays above 0.50. It is not for a lack of correctness, as Figure 7.4 shows, but more that isolates simply are not being clustered as we restrict the `MinPts` value. In fact, the overall clustering purity in Figure 7.5 stays relatively constant. This means that if an isolate is clustered by our algorithm, it will likely be clustered with other isolates of the same host-species.

7.1.6 Discussion

In general, we observe two trends in our data. For the isolates that get clustered into strains, our approach correctly identifies the host-species with over 80-85% accuracy. This accuracy is sufficient to conduct sophisticated MST studies. Most of the strains discovered in the CPLOP data show high degree of purity, and even considering the presence of a few large impure clusters, most of the clustered isolates fall into strains of high purity.

At the same time, the pure strain-based approach suffers from a drop in the coverage as the size of a cluster grows. This means that in general CPLOP isolates tend to be very diverse and come from strains for which not enough DNA material has been collected and pyrosequenced. Identifying the host-species for isolates that do not fall into strains/clusters using the pure strain-based method is impossible. In future work, our goal is to combine the k-NN-based MST method of [36] with the strain-based approach discussed in this paper to increase coverage while preserving the high MST accuracy.

One factor explaining the large impure clusters is the possibility that these clusters represent what the biologists call "transient" strains, i.e., strains that persist in more

than one host-species. Such a characteristic can compound MST by making certain strains of *E. coli* less reliable as FIB for identifying host-species. In Figure 7.2, we see evidence of that and it is revealed in Figure 7.3. One mitigation strategy may be to reduce the presence of these strains in library holding the FIB. Another may be to fall back to an alternative MST technique that works with CPLOP when an unknown isolate falls into an impure cluster. Finally, if a true transient strain is indeed discovered, and an isolate is mapped to it, our MST procedure can simply acknowledge the that the query isolate belongs to a transient strain and provide information about the host-species that show high frequency of *E. coli* incidence from this strain.

7.2 Classifying

7.2.1 Adjusting k

Adjusting k is an important first step. We investigate k values ranging from 1 to 17, but focus primarily on $k \leq 12$. At this point, we do not filter the results in order to focus primarily on the affect of the size of the k-nn list. Thus, α is 0, allowing for the full k list to factor into classification.

Overall, for $k \geq 5$, the accuracy does not improve, but instead levels off. Depending on the resolution algorithm, this value is between 65% and 75% accuracy, as shown in Figure 7.6. By "overall," we mean that for every classification, we validated if it was correct and calculated what proportion to all classifications made that represents to determine accuracy. When looking at all classifications, precision and recall are identical values, as is F-measure.

One good example is the Cow. As Figure 7.7 shows, Cow follows a trend similar to the overall accuracy, staying roughly between 70% and 95% accurate. Certain

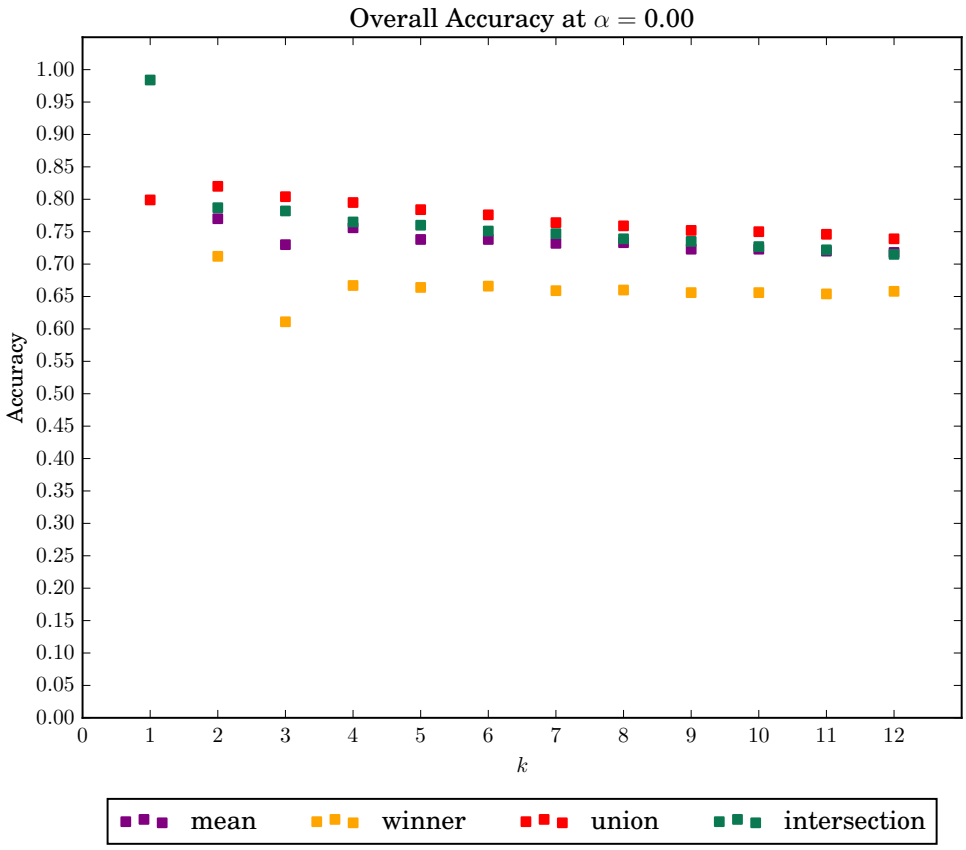


Figure 7.6: The accuracy of all classifications performed with CPLOP across the four different algorithms with $\alpha = 0.00$ shows little improvement for $k > 5$. We look at only the percentage of correct classifications, since that value is equivalent to the precision and the recall.

algorithms get worse for $k > 5$, while other improve.

Figure 7.8 examines the relationship between R and P. This can help us understand the trade offs of choosing one k over another. We will later build a meaningful strategy for how confident we are at recalling a species versus our confidence in a classification of a species.

7.2.2 Adjusting α

By adding a threshold value, we investigated whether this further limitation improves the accuracy by restricting outliers from populating a k-nn list. We investigate

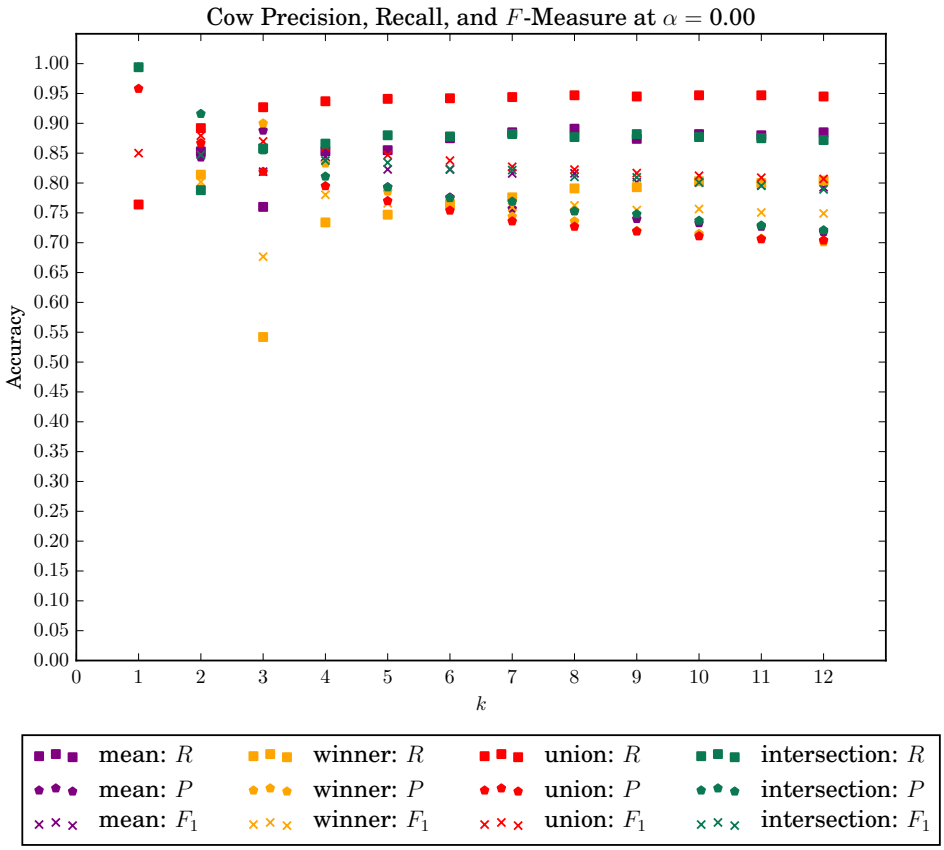


Figure 7.7: There are 1838 Cow isolates in CPLOP. For most resolution algorithms, we observe little improvement when $k > 5$.

$\alpha = \{0.00, 0.98, 0.99\}$. Outside of this study, $\alpha = 0.99$ defines the boundary between strains. One reason we investigate 0.98 is to see whether loosening our definition of strain differentiation gives us a better accuracy.

Overall, we observe that the accuracy slightly improves as we increase the α threshold. Figure 7.9 shows that overall, the accuracy increases as we increase α.

Adding the α made minimal changes to the accuracy of Cow classifications, so only the recall versus precision is shown in Figure 7.10. More details into how α affect the classification accuracy can be seen in Tables 7.1, 7.2, and 7.3.

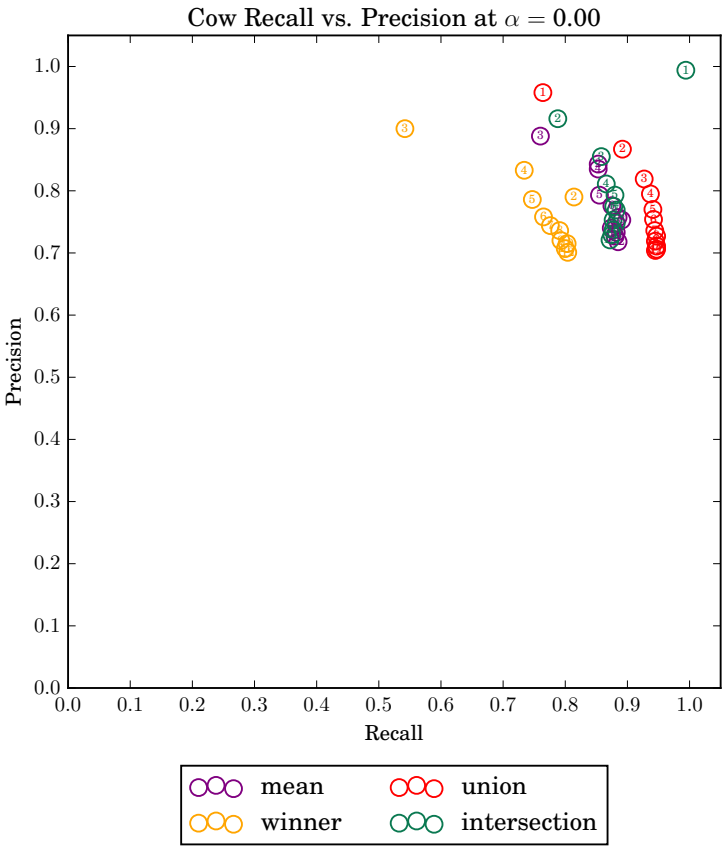


Figure 7.8: There are 1838 Cow isolates in CPLOP. Looking at the Recall as it compares to the Precision for $\alpha = 0.99$ allows us to visualize the tradeoffs we make when picking a k value. Labeled within each datapoint is the k value at that point

7.2.3 Adjusting the Algorithm

Choosing which algorithm to resolve the two different regions of each isolate is an important step. We investigate the differences between the aforementioned four algorithms as they relate to k and α values and how each differ among species of different representation. With library-based-MST, it is important to realize the representation of a species in the library may heavily skew the accuracy of the library.

While interpreting the data, we state that there may be some "%" increase or decrease which we intend to mean the increase in the raw value of the percentage. Additionally, values in the tables represent the proportion of the three metrics, but

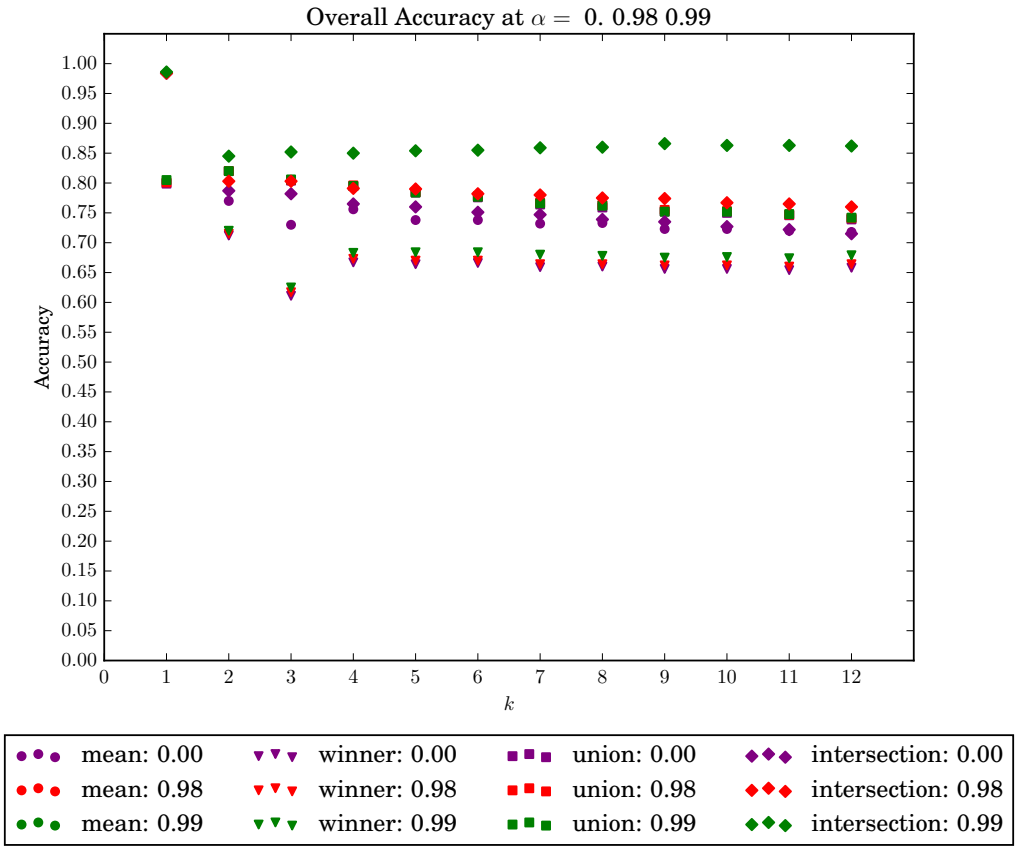


Figure 7.9: Shown is the accuracy of all classifications performed with CPLOP across the four different algorithms. We find that the accuracy of certain resolution algorithms perform better with higher α values.

are easily interpreted as percentages. Section 6 explains the meaning of each precision (P), recall (R), and F-measure (F_1).

Overall, with $\alpha = 0.00$, Figure 7.6 illustrates that the resolution by union algorithm consistently performs better. For $k = 7$ and $\alpha = 0.00$, Table 7.1 shows that using the resolution by unions algorithm performs with 76.4% accuracy with mean-wise and resolution by winner and intersection respectively achieving 73.2%, 65.9%, and 74.7 accuracy%.

Poorly represented species, like the Cat, Chicken, and Seagull did not benefit from the resolution by union algorithm, each achieving no classifications, correct or otherwise.

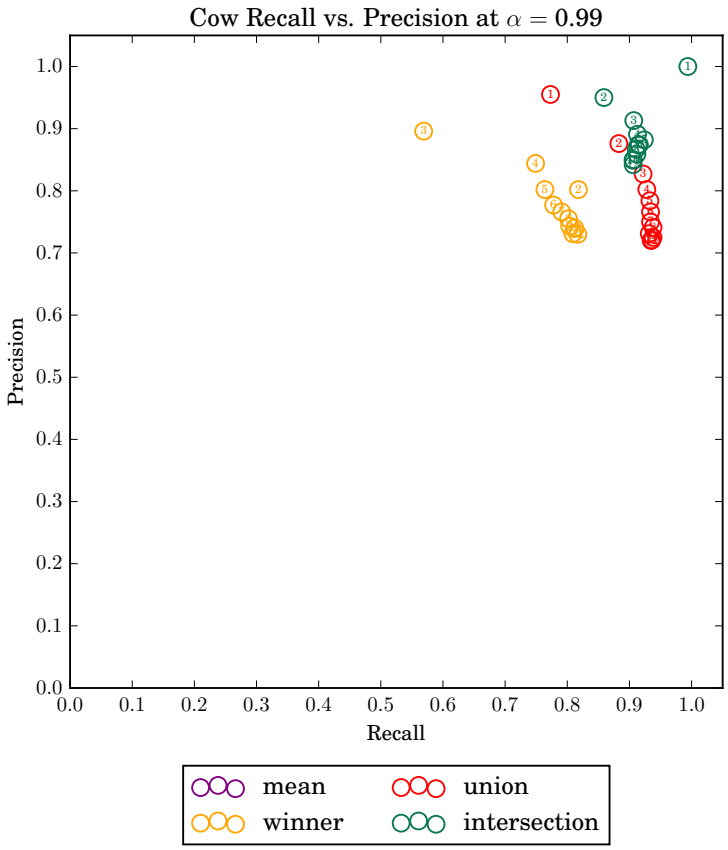


Figure 7.10: There are 1838 Cow isolates in CPLOP. Increasing the α for a species with this many isolates made minimal improvements to the accuracy on all but the resolution by intersection algorithm, which, when compared to Figure 7.8 noticeably improved.

Once we restrict with a somewhat loose threshold of 0.98, overall we see that the intersection method provides the best accuracy, improving on non-thresholded values. For $k = 7$ and $\alpha = 0.98$, the intersection algorithm achieves 78.0% accuracy, while resolution by winner and union respectively achieve 66.4% and 76.7% accuracy.

Table 7.2 shows that a handful of poorly represented species achieved slightly better results when $\alpha = 0.98$. Notably, the intersection algorithm F-measure increased slightly for Wild Turkey, Cat, and Chicken on the order of 3%.

Unfortunately, the meanwise algorithm fails to classify when we use a large enough α and thus we have ommited the results in Tables 7.2 and 7.3. In certain cells of the

Table 7.1: Precision (P), Recall (R), and F-Measure (F_1) overall and for particular species at k=7, $\alpha = 0.00$.

		Meanwise			Winner		
Host-Species	|Isolates|	P	R	F_1	P	R	F_1
Overall	4682	0.732	0.732	0.732	0.659	0.659	0.659
Human	1471	0.857	0.922	0.888	0.771	0.861	0.814
Cow	1718	0.757	0.885	0.816	0.744	0.776	0.760
Pigeon	194	0.420	0.242	0.307	0.280	0.253	0.266
Dog	149	0.596	0.436	0.504	0.449	0.356	0.397
Wild Turkey	72	0.383	0.250	0.303	0.277	0.181	0.219
Chicken	40	0.182	0.050	0.078	0.143	0.075	0.098
Cat	39	0.571	0.308	0.400	0.438	0.359	0.395
Bat	37	0.857	0.973	0.911	0.692	0.973	0.809
Seagull	11	0.000	0.000	0.000	0.000	0.000	0.000

		Union			Intersection		
Host-Species	|Isolates|	P	R	F_1	P	R	F_1
Overall	4682	0.764	0.764	0.764	0.747	0.747	0.747
Human	1471	0.843	0.930	0.884	0.839	0.925	0.880
Cow	1718	0.736	0.944	0.827	0.769	0.882	0.822
Pigeon	194	0.569	0.170	0.262	0.470	0.284	0.354
Dog	149	0.761	0.450	0.566	0.649	0.497	0.563
Wild Turkey	72	0.688	0.306	0.424	0.510	0.361	0.423
Chicken	40	0.000	0.000	0.000	0.250	0.100	0.143
Cat	39	0.889	0.410	0.561	0.571	0.308	0.400
Bat	37	0.857	0.973	0.911	0.857	0.973	0.911
Seagull	11		0.000			0.000	

tables, including Table 7.1, empty values in either P or F_1 mean no classifications were made of that species.

Restricting with $\alpha = 0.99$, our definition of strain differentiation, overall accuracy improves more with resolution by intersection and less so with resolutions by winner and union, garnering 85.9%, 68.0%, and 76.6% accuracy respectively. Again, meanwise resolution fails to produce any classifications.

For poorly represented species, we see some similar improvements for P, R, and F_1, but also some exceptions. Wild Turkey for example, improves by about 2%-3% for resolutions by winner and union and 11% for resolution by intersection, while Cat

Table 7.2: Precision (P), Recall (R), and F-Measure (F_1) overall and for particular species at k=7, $\alpha = 0.98$.

		Winner		
Host-Species	|Isolates|	P	R	F_1
Overall	4682	0.664	0.664	0.664
Human	1471	0.773	0.865	0.816
Cow	1718	0.749	0.777	0.763
Pigeon	194	0.287	0.254	0.269
Dog	149	0.448	0.349	0.392
Wild Turkey	72	0.308	0.222	0.258
Chicken	40	0.150	0.075	0.100
Cat	39	0.467	0.359	0.406
Bat	37	0.692	0.973	0.809
Seagull	11	0.000	0.000	0.000

		Union			Intersection		
Host-Species	|Isolates|	P	R	F_1	P	R	F_1
Overall	4682	0.767	0.767	0.767	0.780	0.780	0.780
Human	1471	0.845	0.930	0.885	0.876	0.950	0.912
Cow	1718	0.742	0.943	0.831	0.799	0.894	0.844
Pigeon	194	0.538	0.181	0.271	0.521	0.333	0.406
Dog	149	0.756	0.456	0.569	0.698	0.536	0.606
Wild Turkey	72	0.697	0.319	0.438	0.571	0.387	0.461
Chicken	40	0.000	0.000	0.000	0.308	0.121	0.174
Cat	39	0.889	0.410	0.561	0.632	0.353	0.453
Bat	37	0.857	0.973	0.911	0.878	0.973	0.923
Seagull	11		0.000		0.000	0.000	0.000

decreases by 3% for resolution by winner, but improves by 6% and 47% for resolution by union and intersection.

7.2.4 Underrepresented Species

Some species had worse accuracy than the overall accuracy. In particular, species such as Chicken with only 40 isolates representing it showed similar leveling of accuracy for $k > 5$, but had far poorer accuracy, as shown in Figure 7.11. For $k > 5$, the accuracy of classifying chicken ranges from as low as 10% to a peak of 26%. The classification accuracy for many species in CPLOP heavily relies on its representation in CPLOP.

Table 7.3: Precision (P), Recall (R), and F-Measure (F_1) overall and for particular species at k=7, $\alpha = 0.99$.

		Winner		
Host-Species	|Isolates|	P	R	F_1
Overall	4682	0.680	0.680	0.680
Human	1471	0.780	0.872	0.823
Cow	1718	0.766	0.791	0.778
Pigeon	194	0.314	0.263	0.286
Dog	149	0.527	0.401	0.455
Wild Turkey	72	0.320	0.222	0.262
Chicken	40	0.167	0.100	0.125
Cat	39	0.433	0.333	0.376
Bat	37	0.720	0.973	0.828
Seagull	11	0.429	0.273	0.334

		Union			Intersection		
Host-Species	|Isolates|	P	R	F_1	P	R	F_1
Overall	4682	0.766	0.766	0.766	0.859	0.859	0.859
Human	1471	0.843	0.925	0.882	0.926	0.979	0.952
Cow	1718	0.750	0.934	0.832	0.874	0.914	0.894
Pigeon	194	0.476	0.205	0.287	0.611	0.468	0.530
Dog	149	0.739	0.456	0.564	0.838	0.738	0.785
Wild Turkey	72	0.719	0.319	0.442	0.667	0.455	0.541
Chicken	40	0.000	0.000	0.000	0.000	0.000	0.000
Cat	39	0.938	0.385	0.546	0.909	0.588	0.714
Bat	37	0.837	0.973	0.900	0.973	1.000	0.986
Seagull	11	0.000	0.000	0.000		0.000	

One notable exception is the Bat. In everyone application of our k-NN algorithms, Bat has above 95% accuracy. It is possible that due to their small size and relative dietary segregation from the surrounding environment that the strains of *E. coli* stay particularly unique. It may also be a quirk of the fact that each isolate comes from a single host-animal, making it difficult to draw conclusions from such results.

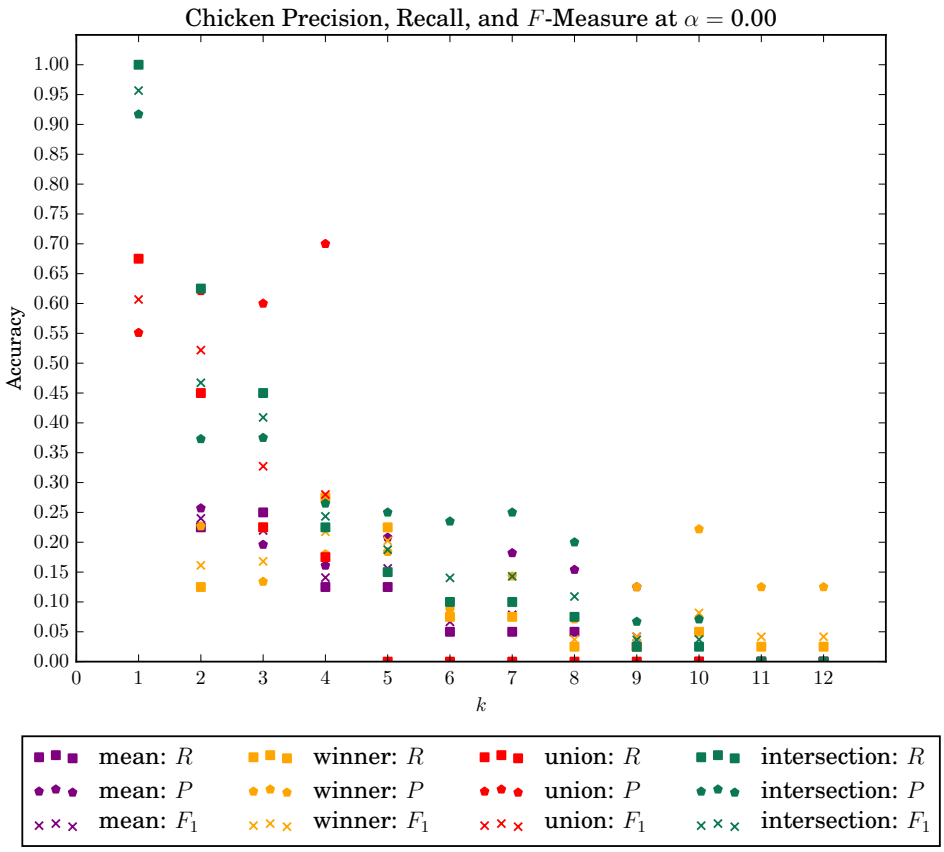


Figure 7.11: There are 40 Chicken isolates in CPLOP. Unfortunately, due to their low representation in CPLOP, classification accuracy is low.

Chapter 8

CONCLUSION

In order to combat the issue of contamination of publicly accessible water supplies, namely fecal contamination, the Cal Poly Biological Sciences Department teamed up with the Cal Poly Computer Science Department to build a library-based MST method, called CPLOP. Using the *E. coli* isolated from fecal samples as fecal indicator bacteria, Cal Poly students pyrosequence the PCR-amplified internal transcribed spacer regions of the *E. coli* and store the resulting vector, called a pyroprint, in the CPLOP database for later retrieval, analysis, and comparison. This thesis investigates two MST methodologies: a density-based clustering method built for CPLOP that clusters for bacterial strains in order to classify an unknown isolate and the k-Nearest Neighbors Resolution Algorithms for Pyroprints (k-RAP), a set of four k-nearest neighbors list resolution strategies for data with multiple comparison functions.

8.1 Clustering for Bacterial Strains

In this paper, we study the accuracy of a clustering-based MST approach which scales significantly better: the bacterial isolate information stored in CPLOP is clustered using an efficient density-based clustering technique. It clusters data by taking two parameters — the minimum number of neighbors and an ε range that those neighbors must be within to form a cluster — and performs range queries in a spatial index that performs $O(\log n)$ look-up on neighbors using a comparison metric. Compared to previous work [36, 41], it requires fewer comparisons to other isolates and computational resources, by being able to perform reasonably fast clusterings on a consumer laptop in minutes.

To ascertain how well this technique classifies, we build a notion of cluster purity. By calculating the proportion of the entire cluster that the most-plural host-species makes up, we hope to understand how a density-based clustering algorithm clusters the CPLOP data. Furthermore, we inspect coverage and overall accuracy.

Results are that we are able to cluster most of the isolates in CPLOP with high accuracy. Most clusters have high purity and low number of unique species, which is promising for using this for MST. Transient strains are also visible in the clustering technique, which will further aid the biologists working on CPLOP in researching transient strains and improving MST techniques. Future work will leverage other MST techniques designed for CPLOP against this to make up for the lack of coverage and transient strains.

8.2 k-RAP Effectiveness

Generally, when using k-NN, it is preferred to use single digit k values. Through our investigation of these various k-NN classification algorithms, we find that that general advice holds true. For our dataset, using $k \geq 5$ does not produce much different results. Choosing $k < 5$ is a dangerous notion, since it is likely that an outlier may make its way into the k-nearest neighbors list, confounding the results. Staying with $5 \leq k \leq 9$ appears to be a safe and reasonable option, providing a good balance between accuracy and filtering of isolates.

Outside of this study, we choose to differentiate between strains of *E. coli* using $\alpha = 0.99$. It appears that using this value is advantageous. There were, however, some exceptions to those results, motivating us to consider non-thresholded k-nearest neighbors lists when classifying an unknown isolate.

The four resolution algorithms — meanwise, winner, union, and intersection — each have their own quirks and behaviors as we alter k and α.

Meanwise, which currently uses the Euclidean norm to resolve different metrics, did not respond to the α threshold and completely stopped classifying anything for α near 1. This is very likely due to Euclidean norm mapping $([0,1],\ldots,[0,1]) \to [0,\sqrt{1+\cdots+1}]$. To get around this, we multiplied the resulting norm by a factor of $\sqrt{2}$, which may have unexpected results. We may investigate this further, or choose a more natural norming method, like arithmetic or geometric mean. With no α filtering, it performed third best with an overall 73.2% classification accuracy.

Winner performs worst, classifying accurately between 65% and 68% of the time. Some alterations to this algorithm may make it more reliable, such as only counting the species that appear in all lists.

Unionwise performs very well. Without filtering the k-nearest neighbors lists by α, we find that the unionwise method classifies best, with an overall accuracy of 76.4%. However, once we add in α filtering, the unionwise does not improve, staying relatively close to 76%.

Intersection performs best when we use α. This is likely due to the "list" actually being a set of common isolates. Overall, without filtering, the accuracy was 74.7%, 78%, and 85.9% for α =0.00, 0.98, and 0.99 respectively.

Overall, we find that the intersection algorithm performs the best and recommend moving forward with it. While unionwise did perform well, it did not respond well to thresholding and still did not perform as well as the intersection algorithm overall. Meanwise and winner may be more useful with previously mentioned modifications and we may investigate these in the future.

Poorly distributed representation of species and environmental incomparabilities are issues endemic to library-based MST. CPLOP has an overabundance of Cow and Human isolates, and an underrepresentation of many of the species in the database. This dilutes the k-nearest neighbors list considerably for species like the Chicken and

Cat.

Library population issues aside, environmental limitations are another concern for accuracy. Nearly every sample in the library comes from a 30 mile radius around Cal Poly, making the collected pyroprints potentially incomparable to pyroprints collected from a different region.

Index

BIBLIOGRAPHY

[1] *2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops, BIBMW 2012, Philadelphia, USA, October 4-7, 2012*. IEEE, 2012.

[2] C. C. Adams. Using Hadoop to Identify False Positives in Bacterial Strain Typing from DNA Fingerprints. *California Polytechnic State University, San Luis Obispo*, 2016.

[3] J. M. Albert, J. Munakata-Marr, L. Tenorio, and R. L. Siegrist. Statistical evaluation of bacterial source tracking data obtained by rep-PCR DNA fingerprinting of Escherichia coli. *Environmental science & technology*, 37(20):4554–4560, 2003.

[4] S. D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In Shavlik [63], pages 37–45.

[5] S. D. Bay. Nearest neighbor classification from multiple feature subsets. *Intell. Data Anal.*, 3(3):191–209, 1999.

[6] L. Belanche-Muñoz and A. R. Blanch. Machine learning methods for microbial source tracking. *Environmental Modelling & Software*, 23(6):741–750, 2008.

[7] N. Bhatia and Vandana. Survey of nearest neighbor techniques. *CoRR*, abs/1007.0085, 2010.

[8] G. Bitton. Microbial indicators of fecal contamination. *Wastewater Microbiology, Third Edition*, pages 153–171, 2005.

[9] M. W. Black, J. VanderKelen, A. Montana, A. Dekhtyar, E. Neal, A. Goodman, and C. L. Kitts. Pyroprinting: A rapid and flexible genotypic

fingerprinting method for typing bacterial strains. *Journal of Microbiological Methods*, 105:121 – 129, 2014.

[10] D. Brandt, A. Montana, B. Somers, M. Black, A. Goodman, and C. Kitts. Pyroprinting sensitivity analysis on the GPU. In 2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops, BIBMW 2012, Philadelphia, USA, October 4-7, 2012 [1], pages 951–953.

[11] Cal Poly. Cal Poly Github, 2016. `http://www.github.com/CalPoly`.

[12] K. R. CLARKE. Non-parametric multivariate analyses of changes in community structure. *Australian journal of ecology*, 18(1):117–143, 1993.

[13] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13(1):21–27, 1967.

[14] T. R. Desmarais, H. M. Solo-Gabriele, and C. J. Palmer. Influence of soil on fecal indicator organisms in a tidally influenced subtropical environment. *Applied and environmental microbiology*, 68(3):1165–1172, 2002.

[15] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Ricci et al. [55], pages 107–144.

[16] J. W. Dickerson Jr. *Evaluation, Development and Improvement of Genotypic, Phenotypic and Chemical Microbial Source Tracking Methods and Application to Fecal Pollution at Virginia's Public Beaches*. PhD thesis, Virginia Polytechnic Institute and State University, 2008.

[17] J. R. Dillard. Demographics and Transfer of Escherichia coli Within Bos taurus Populations. Master's thesis, California Polytechnic State University, San Luis Obispo, 2015.

[18] J. R. Dillard, J. J. VanderKelen, J. D. Kent, A. D. Frey, P. J. McCreesh, D. Britton, T. Branck, M. W. Black, and C. L. Kitts. E. coli Strain Demographics and Transmission in Cattle. *Strain*, 10(1):11, 2013.

[19] W. Ding, T. Washio, H. Xiong, G. Karypis, B. M. Thuraisingham, D. J. Cook, and X. Wu, editors. *13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013*. IEEE Computer Society, 2013.

[20] P. B. Eckburg, E. M. Bik, C. N. Bernstein, E. Purdom, L. Dethlefsen, M. Sargent, S. R. Gill, K. E. Nelson, and D. A. Relman. Diversity of the human intestinal microbial flora. *science*, 308(5728):1635–1638, 2005.

[21] 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.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data MiningI*, pages 226–231. AAAI Press, 1996.

[23] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[24] E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

[25] E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: Small sample performance. Technical report, DTIC Document, 1952.

[26] Q. Hua, A. Ji, and Q. He. Multiple real-valued K nearest neighbor classifiers system by feature grouping. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, 10-13 October 2010 [51], pages 3922–3925.

[27] J. Huan, S. Miyano, A. Shehu, X. T. Hu, B. Ma, S. Rajasekaran, V. K. Gombar, M. Schapranow, I. Yoo, J. Zhou, B. Chen, V. Pai, and B. G. Pierce, editors. *2015 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2015, Washington, DC, USA, November 9-12, 2015*. IEEE Computer Society, 2015.

[28] *International Conference on Machine Learning and Cybernetics, ICMLC 2010, Qingdao, China, July 11-14, 2010, Proceedings*. IEEE, 2010.

[29] L. Jiang, Z. Cai, D. Wang, and S. Jiang. Survey of improving k-nearest-neighbor for classification. In Lei [34], pages 679–683.

[30] E. Johnson. Density-Based Clustering of High-Dimensional DNA Fingerprints for Library-Dependent Microbial Source Tracking. Master's thesis, California Polytechnic State University, San Luis Obispo, 2015.

[31] W. Juan. Multiple nearest neighbor classifiers system based on feature perturbation by mutual information. In International Conference on Machine Learning and Cybernetics, ICMLC 2010, Qingdao, China, July 11-14, 2010, Proceedings [28], pages 247–251.

[32] J. Kent, M. Alvarado, J. VanderKelen, A. Montana, J. Soliman, A. Dekhtyar, A. Goodman, C. Kitts, and M. Black. Pyroprinting: Novel Pyrosequencing-Based Method for Studying E. coli Diversity and Microbial Source Tracking (779.8). *The FASEB Journal*, 28(1 Supplement):779–8, 2014.

[33] D. T. Larose. *Discovering knowledge in data: an introduction to data mining.* John Wiley & Sons, 2005.

[34] J. Lei, editor. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007, 24-27 August 2007, Haikou, Hainan, China, Proceedings, Volume 1.* IEEE Computer Society, 2007.

[35] W. Li, D. Raoult, and P.-E. Fournier. Bacterial strain typing in the genomic era. *FEMS Microbiology Reviews*, 33(5):892–916, 2009.

[36] J. D. McGovern, A. Dekhtyar, C. Kitts, M. Black, J. Vanderkelen, and A. Goodman. Leveraging the k-nearest neighbors classification algorithm for microbial source tracking using a bacterial DNA fingerprint library. In Huan et al. [27], pages 1694–1701.

[37] J. D. McGovern, E. Johnson, A. Dekhtyar, M. Black, C. Kitts, and J. Vanderkelen. Library-based microbial source tracking via strain identification. In Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2016, Seattle, WA, USA, October 2-5, 2016 [50], pages 364–373.

[38] R. N. McLeod. A Proof of Concept for Crowdsourcing Color Perception Experiments. *California Polytechnic State University, San Luis Obispo*, 2014.

[39] D. J. Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer.* Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[40] A. Montana. Algorithms for Library-Based Microbial Source Tracking. Master's thesis, California Polytechnic State University San Luis Obispo, 2013.

[41] A. Montana, A. Dekhtyar, M. Black, C. Kitts, and A. Goodman. Ontological hierarchical clustering for library-based microbial source tracking. In Ding et al. [19], pages 568–576.

[42] A. Montana, A. Dekhtyar, E. Neal, M. Black, and C. Kitts. Chronology-sensitive hierarchical clustering of pyrosequenced DNA samples of e. coli: A case study. In Wu et al. [72], pages 155–159.

[43] A. Montana, A. Dekhtyar, E. Neal, M. Black, and C. Kitts. Investigating temporal strain diversity in human e. coli populations using pyroprinting: A novel strain identification method. Technical report, Technical report, California Polytechnic State University, San Luis Obispo, CA, 2012.

[44] C. Moritz, D. Shapiro, and C. Pann. Application of Pyroprinting for Source Tracking of E. coli in Pennington Creek. *California Polytechnic State University, San Luis Obispo*, 2015.

[45] E. Neal, C. Sabatini, W. Tang, M. Black, and C. Kitts. Demographics of E. coli Strains in the Human Gut Using Pyroprints: A Novel MST Method. In *CSUPERB, Poster*. Jan, 2012.

[46] E. R. Neal. Escherichia coli Strain Diversity in Humans: Effects of Sampling Effort and Methodology. Master's thesis, California Polytechnic State University, San Luis Obispo, 2013.

[47] M. Neave, H. Luter, A. Padovan, S. Townsend, X. Schobben, and K. Gibb. Multiple approaches to microbial source tracking in tropical northern australia. *MicrobiologyOpen*, 3(6):860–874, 2014.

[48] J. Nguyen, J. Vanderkelen, M. Black, and C. Kitts. Investigating the Dominant Escherichia coli Strain in Lambs and Ewes Using Pyroprinting: A Novel

Method for Strain Identification. *California Polytechnic State University, San Luis Obispo*, 2015.

[49] X. Ning, C. Desrosiers, and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Ricci et al. [54], pages 37–76.

[50] *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2016, Seattle, WA, USA, October 2-5, 2016*. ACM, 2016.

[51] *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, 10-13 October 2010*. IEEE, 2010.

[52] V. Ramachandran, editor. *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas*. ACM/SIAM, 1993.

[53] S. Ranka, iTamer Kahveci, and M. Singh, editors. *ACM International Conference on Bioinformatics, Computational Biology and Biomedicine, BCB' 12, Orlando, FL, USA - October 08 - 10, 2012*. ACM, 2012.

[54] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[55] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[56] C. Ricketts. Cal Poly Library of Pyroprints: Quality Control Analysis and Web Development. Master's thesis, California Polytechnic State University, San Luis Obispo, 2014.

[57] K. Ritter, E. Carruthers, C. Carson, R. Ellender, V. Harwood, K. Kingsley, C. Nakatsu, M. Sadowsky, B. Shear, B. West, et al. Assessment of statistical

methods used in library-based approaches to microbial source tracking. *J Water Health*, 1:209–223, 2003.

[58] S. Rogers and J. Haines. Detecting and mitigating the environmental impact of fecal pathogens originating from confined animal feeding operations: review. *United States Environmental Protection Agency, Office of Research and Development, National Risk Management Research Laboratory*, 2005.

[59] M. Ronaghi, M. Uhlén, and P. Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281(5375):363–365, 1998.

[60] D. Sargeant, W. R. Kammin, and S. Collyard. *Review and critique of current microbial source tracking (mst) techniques*. Environmental Assessment Program, Washington State Department of Ecology, 2011.

[61] T. M. Scott, J. B. Rose, T. M. Jenkins, S. R. Farrah, and J. Lukasik. Microbial source tracking: Current methodology and future directions. *APPLIED AND ENVIRONMENTAL MICROBIOLOGY*, pages 5796–5803, 2002.

[62] D. Shapiro, J. Kent, M. Zuleta, C. Kitts, M. Black, and J. VanderKelen. Source Tracking of Fecal Contamination Along San Luis Obispo (SLO) Creek. *The FASEB Journal*, 29(1 Supplement):575–12, 2015.

[63] J. W. Shavlik, editor. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*. Morgan Kaufmann, 1998.

[64] D. Shealy. Exploration of PyroPrinting for Environmental Forensics. Technical report, California Polytechnic State University, San Luis Obispo, California, June 2012.

[65] J. M. Simpson, J. W. Santo Domingo, and D. J. Reasoner. Microbial source tracking: state of the science. *Environmental science & technology*, 36(24):5279–5288, 2002.

[66] J. L. Soliman. CPLOP: The Cal Poly Library of Pyroprints. Master's thesis, California Polytechnic State University San Luis Obispo, 2013.

[67] J. L. Soliman, A. Dekhtyar, J. Vanderkellen, A. Montana, M. Black, E. Neal, K. Webb, C. Kitts, and A. Goodman. Microbial source tracking by molecular fingerprinting. In Ranka et al. [53], pages 617–619.

[68] J. Stewart, R. Ellender, J. Gooch, S. Jiang, S. Myoda, and S. Weisberg. Recommendations for microbial source tracking: lessons from a methods comparison study. *J Water Health*, 1:225–231, 2003.

[69] J. J. VanderKelen, R. D. Mitchell, A. Laubscher, M. W. Black, A. L. Goodman, A. K. Montana, A. M. Dekhtyar, R. Jimenez-Flores, and C. L. Kitts. Short Communication: Typing and Tracking Bacillaceae in Raw Milk and Milk Powder Using Pyroprinting. *Journal of Dairy Science*, 99(1):146–151, 2016.

[70] L. Wang, Q. Hua, X. Wang, and Q. Chen. Combination of multiple nearest neighbor classifiers based on feature subset clustering method. In Yeung et al. [73], pages 538–547.

[71] K. Webb. Cplop-cal poly's library of pyroprints. *California Polytechnic State University, San Luis Obispo*, 2011.

[72] F. Wu, M. J. Zaki, S. Morishita, Y. Pan, S. Wong, A. Christianson, and X. Hu, editors. *IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2011, Atlanta, GA, USA, November 12-15, , 2011*. IEEE Computer Society, 2011.

[73] D. S. Yeung, Z. Liu, X. Wang, and H. Yan, editors. *Advances in Machine Learning and Cybernetics, 4th International Conference, ICMLC 2005, Guangzhou, China, August 18-21, 2005, Revised Selected Papers*, volume 3930 of *Lecture Notes in Computer Science*. Springer, 2006.

[74] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In Ramachandran [52], pages 311–321.