ENHANCEMENTS TO THE MICROBIAL SOURCE TRACKING PROCESS
THROUGH THE UTILIZATION OF CLUSTERING AND K-NEAREST
CLUSTERS ALGORITHM

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

Tram Lai

March 2018

© 2018
Tram Lai
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Enhancements to the Microbial Source
Tracking Process through the Utilization
of Clustering and k-Nearest Clusters Algo-
rithm

AUTHOR: Tram Lai

DATE SUBMITTED: March 2018

COMMITTEE CHAIR: Alexander Dekhtyar, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Lubomir Stanchev, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Chris Kitts, Ph.D.
Professor of Biological Sciences

ABSTRACT

Enhancements to the Microbial Source Tracking Process through the Utilization of Clustering and k-Nearest Clusters Algorithm

Tram Lai

Bacterial contamination in water sources is a serious health risk and the sources of the bacterial strains must be identified to keep people safe. This project is the result of a collaboration effort at Cal Poly to develop a new library-dependent *Microbial Source Tracking* method for determining sources of fecal contamination in the environment. The library used in this study is called Cal Poly Library of Pyroprints (CPLOP). The process of building CPLOP requires students to collect fecal samples from a multitude of sources in the San Luis Obispo area. A novel method developed by the biologists at Cal Poly called *pyroprinting* is then applied on the two intergenic regions of the *E. coli* isolates from these samples to obtain their fingerprints. These fingerprints are stored in the CPLOP database. In our study, we consider any *E. coli* samples whose fingerprints match above a certain threshold to be in the same group of bacterial strain. However, there has not yet been a final MST method that produces an acceptable level of accuracy. In this thesis, we propose a two-step MST classifier that combines two previous works: pyro-DBSCAN and k-RAP. These algorithms were developed specifically for CPLOP. We call our classifier HAP - Hybrid Algorithm for Pyroprints. The classifier works as follows. Given an unknown isolate, the first step requires performing clustering on the known isolates in the library and comparing the unknown isolate against the resulting clusters. If the isolate falls into a cluster, its classification will be returned as the dominant species of that cluster. Otherwise, we apply the k-Nearest Clusters Algorithm on this isolate to determine its final classification. Ultimately, HAP provides us a set of 16 decision strategies that identify the host species of an unknown sample with high accuracy.

ACKNOWLEDGMENTS

Thanks to:

- My family for their sacrifices, love and support
- Josh Terry for all his love, support and encouragement
- Alex Dekhtyar for his advice, support and accommodation during my years at Cal Poly
- All the faculty in the CS department for their dedication to the students
- Ginger DeWitte, Vivian Fong, Di Hoang, Mike Ryu and other friends who had made my time at Cal Poly memorable

TABLE OF CONTENTS

	Page
LIST OF TABLES	ix
LIST OF FIGURES	xi
CHAPTER	
1 Introduction	1
2 Background & Related Work	6
2.1 Background	6
2.1.1 Microbial Source Tracking (MST)	6
2.1.2 Pyroprinting	8
2.1.3 Internal Transcribed Spacers (ITS)	8
2.1.4 Pyroprint	10
2.1.5 Pearson Correlation	11
2.2 Related Work	13
2.2.1 Cal Poly Library of Pyroprints - CPLOP	13
2.2.2 Ontology-based Hierarchical Clustering - OHClust!	15
2.2.3 Density-Based Spatial Clustering of Applications with Noise - DBSCAN	17
2.2.4 k-Nearest Neighbors Resolution Algorithms for Pyroprints - k-RAP	19
3 Design	22
3.1 Motivation and Goal	22
3.2 Methodology	22
3.2.1 Comparing Isolates	22
3.2.2 α -filtering	23
3.2.3 The Two-step Classification Technique - HAP	24
3.3 Clustering Data	29
3.3.1 Cluster	30
3.3.2 ClusterDataset	30
3.4 k-Nearest Clusters	30
3.4.1 Weighting Scheme	31

3.4.2	Modifications to Meanwise Resolution	35
3.4.3	Modifications to Winner Resolution	38
3.4.4	Modifications to Union Resolution	40
3.4.5	Modifications to Intersection Resolution	42
4	Implementation	46
4.1	Language and Libraries	46
4.2	Overview	47
4.2.1	k-RAP	47
4.2.2	k-Nearest Clusters	47
5	Evaluation	49
5.1	Entropy	50
5.2	minPts & k	51
5.2.1	Narrowing minPts for pyro-DBSCAN	51
5.2.2	Narrowing k for k-Nearest Clusters	53
5.3	Cross-validation with Holdout	55
5.3.1	Precision	58
5.3.2	Recall	59
5.3.3	F-score	59
5.4	Results & Discussion	60
5.4.1	k-Nearest Neighbors	60
5.4.2	Clustering & k-Nearest Neighbors	64
5.4.3	Clustering & k-Nearest Clusters	64
5.4.4	Major & Under-Represented Species	72
6	Conclusion	79
6.1	Clustering Entropy	79
6.2	Accuracy and Effectiveness of HAP	80
6.3	Future Work	81
6.3.1	Incorporate HAP into CPLOP	81
6.3.2	Amazon Web Services - AWS	82
	BIBLIOGRAPHY	83
	APPENDICES	
A	Purity	85

B	HAP - Hybrid Algorithm for Pyroprints	87
B.1	<code>minPts = 4</code>	87
B.2	<code>minPts = 5</code>	87

LIST OF TABLES

Table		Page
2.1	The number of isolates per host species for the top 10 host species in CPLOP	14
3.1	A simple example describing our weighting scheme. Even though cluster-1 is further from the unknown than isolate-1, cluster-1 has a radius of 2, meaning the furthest point from the cluster is only 0.5 apart from the unknown. That means the classification of cluster-1 should be weighted more than isolate-1. Given our weighting scheme, we can calculate the classification for the unknown isolate as such: Human = $1/1.5 = 0.67$; Cat = $2/2.5 = 0.8$. Therefore, the final classification for the unknown isolate is “Cat.”	35
5.1	The weighted entropy and weighted purity of clusters obtained from pyro-DBSCAN with <code>minPts</code> vary from 1 to 9. We do not want to examine any <code>minPts</code> > 9 because of the large decrease in the number of isolates that get clustered. The detail on how to calculate the weighted purity can be found in the Appendix A.	52
5.2	The number of isolates per host species filtered from CPLOP that we considered in this study.	56
5.3	A comparison of McGovern’s results (colored in green) and our results when performing k-RAP for k ranging from 1 to 10 and $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995 . McGovern’s results are taken from chapter 6 in [8]. For our results, in addition to the overall precision, we also calculate the average accuracy out of total isolates, which includes isolates that do not get classified. An empty cell means there is no available data for that particular pair of resolution and α . . .	61
5.4	The number of isolates that are returned as “Undefined” by our modified k-RAP for $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995	61
5.5	The number of isolates that are returned as “Undefined” by our classifier for <code>minPts</code> = 3 and $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995 . .	67
5.6	Precision (P), Recall (R), and F -Measure (F_1) of some particular species in CPLOP obtained from Unweighted Dominant Species selection method (DS) with four resolutions at $k = 7$, $\alpha = 0.00$	74
5.7	Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Unweighted Collective Species selection method (CS) with four resolutions at $k = 7$, $\alpha = 0.00$	75

5.8	Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Weighted Dominant Species selection method (wDS) with four resolutions at $k = 7$, $\alpha = 0.00$	76
5.9	Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Weighted Collective Species selection method (wCS) with four resolutions at $k = 7$, $\alpha = 0.00$	77

LIST OF FIGURES

Figure	Page
2.1 A sample pyrogram - the result of pyrosequencing. The values represent the heights of the peaks	9
2.2 A simplified diagram of <i>E. coli</i> genome, where there are seven copies of rRNA operon. Each rRNA operon contains the two ITS regions of interest, ITS-1 and ITS-2.	10
2.3 A simple dendrogram (tree) representing clusters for a dataset of 5 data points. The process of merging starts at the bottom nodes. From the original 5 nodes 1, 2, 3, 4, and 5, three clusters were formed (1,2) = (6), (3) and (4,5) = (7). Then the clusters (6) combined with (3) to form (8). Finally, cluster (8) merged with (7) to form cluster (9).	16
2.4 An example of a density-based cluster where <code>minPts</code> = 3. Starting at the point in the center, it is a core point since it has at least 3 neighboring points. These neighboring points are also core points since they also have at least 3 neighboring points. Expanding from these core points, the points in green are within ε -neighborhood of core points but they do not have at least 3 neighboring points. They are then classified as border points. Some of the points in grey are within ε -neighborhood of border points but not core points. They also do not have enough neighboring points, hence, they are classified as noise.	18
3.1 A simple example describing how the k-Nearest Clusters Algorithm works when $k = 3$. Given a dataset consisting of 3 clusters, 2 outliers and an unknown point. In order to determine the classification of the unknown point, we look at its three nearest neighboring clusters or points. Assuming no weighting scheme is applied and the classification of each cluster is its dominant label, then the unknown point has two neighbors labeled A and one neighbor labeled B, hence, its classification is A. However, the cluster labeled B contains more points than A, so it is not fair to treat it equally as other clusters. Also, if we look at the composition of each of the unknown point's neighbors, the unknown point is neighbored with a total of 11 points, of which, 4 points labeled A and 7 points labeled B. This means the classification of the unknown can also be B. For this reason, we propose different strategies for our k-Nearest Clusters by varying the weight coefficient and the clusters' composition. We also evaluate each strategy to determine which one works best.	32

3.2	A simple graph annotating isolates in CPLOP in two-dimensional space where the dimensions are the ITS-1 and ITS-2 regions. As previously mentioned, each isolate in CPLOP can be represented as: $I = (\vec{p}_1, \vec{p}_2)$ where $\vec{p}_1$ and $\vec{p}_2$ are the pyroprint vectors of isolate I for the ITS-1 and ITS-2 regions, respectively. From this graph, we can see that when $k = 3$, for the unknown isolate, its three nearest neighbors for the ITS-1 region are [4,1,7] and for the ITS-2 region are [3,2,1].	33
3.3	A simple example of classifying an unknown isolate u using Meanwise Resolution with $k = 5$	37
3.4	A simple example of classifying an unknown isolate u using Winner Resolution with $k = 5$. The “winner” species for ITS-1 is “Dog” with the max count of 3. In ITS-2, “Bat” and “Cat” both have the max count of 2. The algorithm randomly picks the first one in the list as its classification, assumingly in this case “Cat.” The overall “winning” species is “Dog” with the max count of 3. When taking into account the weight coefficient, the final counts might be different but the same concept applied.	39
3.5	A simple example of classifying an unknown isolate u using Union Resolution with $k = 5$. Even though in ITS-1, “Dog” is the most dominant species. In ITS-2, “Cat” has the maximum count. The resulting union set shows “Cat” as the final classification since it has the highest combined count overall. When taking into account the weight coefficient, the final counts might be different but the same concept applied.	41
3.6	A simple example of classifying an unknown isolate u using Intersection Resolution with $k = 5$. The k -NN lists of size 5 of both ITS-1 and ITS-2 only yield 3 isolates in common. Since we want an intersection of size 5, we must keep expanding the two k -NN lists. The intersection set is achieved with when we extend the k -NN lists to size 8. The final classification is Dog, with the max count of 2. . . .	44

5.1	A simple example showing how to calculate a cluster's entropy and the overall entropy. The entropy for the individual cluster can be calculated as showed: $Entropy_1 = -\frac{31}{32} * \log_2(\frac{31}{32}) - \frac{1}{32} * \log_2(\frac{1}{32}) = 0.2$ $Entropy_2 = -\frac{5}{5} * \log_2(\frac{5}{5}) = 0$ $Entropy_3 = -\frac{10}{13} * \log_2(\frac{10}{13}) - \frac{1}{13} * \log_2(\frac{1}{13}) - \frac{2}{13} * \log_2(\frac{2}{13}) = 0.991$ $Entropy_{total} = 0.2 * \frac{32}{50} + 0.991 * \frac{13}{50} = 0.385$	51
5.2	The overall precision of all classifications performed against CPLOP across 4 different resolution algorithms with $\alpha = 0.0$. The accuracy levels off for $k \geq 5$. It reinforces our decision of staying with $5 \leq k \leq 9$.	54
5.3	The overall precision and overall accuracy obtained from our modified k-RAP at $\alpha = 0.95, 0.98, 0.99$ and 0.995 . The overall accuracy values are labeled as "Adjusted."	62
5.4	The overall accuracy when we combined pyro-DBSCAN and k-RAP at minPts =3, $\alpha = 0.00$.	65
5.5	The overall accuracy for four HAP methods with different resolutions at minPts =3 and $\alpha = 0.00$.	68
5.6	The overall precision and accuracy for four HAP methods with different resolutions at minPts =3 and $\alpha = 0.98$.	69
5.7	The overall precision and accuracy for four HAP methods with different resolutions at minPts =3 and $\alpha = 0.99$.	70
5.8	The overall precision and accuracy for four HAP methods with different resolutions at minPts =3 and $\alpha = 0.995$.	71
5.9	The overall accuracy for four <i>strict</i> -HAP methods with different resolutions at minPts = 3, $\alpha = 0.00$.	73

A.1	A simple example showing how to calculate the clustering purity. $purity_1 = \max(\frac{31}{32}, \frac{1}{32}) = \frac{31}{32} = 0.969$ $purity_2 = \max(\frac{5}{5}) = 1$ $purity_3 = \max(\frac{10}{13}, \frac{1}{13}, \frac{2}{13}) = \frac{10}{13} = 0.769$ $Entropy_{total} = 0.969 * \frac{32}{50} + 1 * \frac{5}{50} + 0.769 * \frac{13}{50} = 0.92$	86
B.1	The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.00$	88
B.2	The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.98$	89
B.3	The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.99$	90
B.4	The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.995$	91
B.5	The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.00$	92
B.6	The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.98$	93
B.7	The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.99$	94
B.8	The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.995$	95

Chapter 1

INTRODUCTION

Bacterial contamination in any human consumption products, especially in water and food, can have a tremendous impact not only on human health but also on society as a whole and the future of agriculture. In 1998, it was estimated that 2.2 million deaths were associated with diarrhea yearly, a majority of these due to fecal pollution of water [10]. Studies done by the Food and Agricultural Organization (FAO) found that “domestic animals such as poultry, cattle, sheep, and pigs generate 85% of the world’s animal faecal waste.” This rate amounts to about 2.62×10^{13} kg/year [5]. Considering that the world population keeps growing every year, the demand for animal products also rises as a result. It is no longer sufficient to just control the contamination, many Environmental and Health organizations have become more interested in finding the source of the contamination. Identifying bacterial sources is an important first step not only in controlling fecal contamination, but also in removing and reducing the environmental risks.

Microbial Source Tracking (MST) refers to a collection of methods (microbiological, genotypical, phenotypical and chemical) used to determine the source of fecal contamination in the environment [12]. These methods often look for *fecal indicator bacteria* (FIB) in the environmental samples and make an attempt to ascertain the sources of fecal pollution. Currently, all MST techniques are experimental and it is practically impossible to identify all possible fecal sources accurately. In our study, we are not so much interested in the question of “What is the exact animal that the fecal matter came from?” but more in the overarching question of “What is the host species or subspecies that the fecal matter likely came from?”

Pyroprinting is a novel method developed at Cal Poly as an ongoing collaboration

effort by the faculty and students in the Biology, Chemistry, and Computer Science departments [4]. It is based on a DNA sequencing method called *pyrosequencing*. The results of pyroprinting are digital fingerprints called *pyroprints*. They are stored in the Cal Poly Library of Pyroprints (CPLOP) to later be used in our MST process. CPLOP was started in 2011 as a quarter-long class project with the motivation of making the MST process cheaper, faster, more reliable, and more discriminating. The process of building CPLOP requires students to collect fecal samples from several known host species in the San Luis Obispo area, run pyroprinting on the two intergenic regions of the rRNA genes of the *Escherichia coli* (E. coli) isolates [14], and finally, store the results in CPLOP.

The original purpose of CPLOP was to store and manage pyroprints. However, more work was needed because determining whether two pyroprints are similar is not sufficient to conclude that two samples of FIB came from the same host species. This is where the concept of a strain comes in. Our general definition of *a strain* is a group of bacteria that are closely related, determined by a metric of similarity and a threshold. Once the concept of strain was defined, many studies were conducted to expand on this idea. In 2013, CPLOP version 2.0 was developed and released. The new version of CPLOP contains many new functions, such as recognizing groups of a similar strain, supporting strain matching for isolates, and flagging and removing erroneous pyroprints [14]. Many algorithms, such as OHClust!, pyro-DBSCAN and k-RAP, were later designed and developed specifically for CPLOP [9, 7, 8]. These algorithms not only leverage CPLOP data to analyze bacterial fingerprints but also use these data to validate the classification resulting from MST.

OHClust! was the first clustering algorithm developed for CPLOP. It is based on agglomerative hierarchical clustering, utilizing the information recorded for each sample to allow an efficient addition to known clusters through incremental update [9]. Even though OHClust! did not succeed in determining if the clusters met the

researchers’ definition of strains, it set up the direction for the work that followed. In 2015, Johnson developed an algorithm to use as an alternative clustering algorithm to replace OHClust! in CPLOP [7]. Johnson’s algorithm is a modification of the original DBSCAN, a density-based clustering algorithm, that can identify *E. coli* strains from pyroprints. In order to differentiate Johnson’s algorithm from the original DBSCAN, hereafter Johnson’s algorithm shall be referred to as pyro-DBSCAN. Compared to OHClust!, pyro-DBSCAN is not only faster, but also more effective in finding clusters that are likely from similar strains. However, when incorporating pyro-DBSCAN into CPLOP, not all isolates get clustered. This issue prompted the development of k-RAP, which is a fallback method for when pyro-DBSCAN fails to make a determination [8]. k-RAP is a modification of k-Nearest Neighbors algorithm with different resolution methods proposed to accommodate our MST process.

A lot of effort has been put into supporting MST, however, there has not been a final method proposed for MST with an acceptable level of accuracy. Our current outsourced method, which is to send samples to an external laboratory for analysis, produces an accuracy of around 30%. Using pyro-DBSCAN with the minimum neighbors ranging from 1 to 9, we can ensure that at least 40% of the isolates get clustered and the algorithm can correctly identify the host species 80% of the time. Investigating k-RAP showed that for the entire CPLOP database, the host species can be accurately identified above 50% of the time with most of the resolution algorithms. Seeing the success of these two algorithms individually, we attempt to build an effective classification technique by combining clustering using pyro-DBSCAN and k-RAP. In this paper, we propose HAP, a two-step classification technique, that stands for Hybrid Algorithm for Pyroprints.

In HAP, the pre-processing step involves clustering all known isolates in the library. Then, given an unknown isolate, the first step requires checking this isolate against the clusters that are created using pyro-DBSCAN. An isolate belongs to a

cluster if its distance to the cluster’s centroid is within the cluster’s radius (more details can be found in Chapter 3). We break ties by selecting the cluster whose distance to the isolate is the smallest. If the unknown isolate is found to belong to a cluster, its classification will be returned as the dominant species of that cluster. If the unknown isolate does not belong to any cluster, we proceed to the second step where we perform k-Nearest Clusters. For this step, we also propose four methods to manage clusters’ properties:

- **Unweighted Dominant Species selection (DS) method:** Performs k-nearest clusters where we treat each cluster equally and we consider each cluster as a single isolate whose host species is the dominant species of the cluster.
- **Unweighted Collective Species selection (CS) method:** Performs k-nearest clusters where we treat each cluster equally and we consider each cluster as a single isolate whose host species is a collection/group of all species in that cluster.
- **Weighted Dominant Species selection (wDS) method:** Repeats DS method with the addition of a cluster weight coefficient (more details can be found in Chapter 3).
- **Weighted Collective Species selection (wCS) method:** Repeats CS method with the addition of a cluster weight coefficient.

k-RAP is also applied in each of these methods to resolve multiple k-NN lists into a single list. In the case where there are several host species in the final list, the algorithm randomly selects the first one in the list.

Evaluating HAP showed us that the classification accuracy for the entire CPLOP database stayed well above 65% for most pairs of HAP methods and k-RAP resolutions. These results are not only far better than our outsourced method, but they

are also better than what was previously obtained by pyro-DBSCAN and k-RAP individually. The results show that the DS method performs better than CS method, while the DS and CS methods perform better than their weighted counterparts.

This thesis is an extension of the MST methodologies developed by Johnson and McGovern [7, 8]. The contributions are the following:

- A set of different decision strategies through the combination of the clustering algorithm (pyro-DBSCAN) and k-RAP to better identify the host species of an unknown sample,
- An empirical study measuring the accuracy of identifying the host species for the samples stored in CPLOP using HAP,
- An evaluation on the accuracy of HAP compared to the individual accuracy of clustering (pyro-DBSCAN) and k-RAP, and
- An investigation of the clustering algorithm (pyro-DBSCAN) to measure the overall clustering entropy.

The rest of this document is organized as follows. Chapter 2 provides the background information about the biology and computer science sides of the problem we are trying to solve. In addition, this chapter also provides information on previous works. Chapter 3 outlines the design for our solution. Chapter 4 provides the solution implementation. Chapter 5 describes the evaluations performed on the solution. Finally, Chapter 6 concludes this thesis and suggests some ideas for future work.

Chapter 2

BACKGROUND & RELATED WORK

This chapter provides the background information and some previous studies that were done in an attempt to improve the MST process. These studies involve the improvement of the pryoprint library - CPLOP, different clustering methods, such as OHClust! and pyro-DBSCAN, and k-RAP, a modification of k-Nearest Neighbors with restricted k values and a threshold to resolve multiple k-NN lists [14, 9, 7, 8].

2.1 Background

2.1.1 Microbial Source Tracking (MST)

Microbial Source Tracking (MST) is a field of study aimed to help identify the host species from which an unknown microbial sample originated. There are several different approaches to MST. One of them is library-independent methods, which requires no culturing or isolation of individual bacteria. Even though this technique sounds easier, it unfortunately relies on prior knowledge of source-specific pathogens, hence limiting the effectiveness of host species determination. Another approach is library-based methods, which uses a library of known species and works off of an assumption that there are enough genetic differences in the indicator organisms that can be used to differentiate between species [12]. In order for library-based methods to be effective, the researchers have to collect a significant number of samples from known host species, run genetic typing methods on them, and then use this data to determine a host species of an unknown sample. Even though this method might be more expensive initially because it requires a library with a minimum of 2000-3000 isolates in order to be effective, in the long run, researchers can use it to test any sample from

a variety of host species [1].

Indicator organisms play an important role in MST. Fecal indicator bacteria (FIB) are micro-organisms that inhabit the gastrointestinal tract of many hosts. They are used to predict the presence of pathogens. A pathogen is an agent that can cause diseases in living organisms. Indicator organisms are useful in that they are not necessarily harmful but are associated specifically with feces and are easy to detect. Using fingerprints from FIB in MST can help us in determining what host species an unknown sample comes from. A few common indicators for MST are *Escherichia coli*, and *Enterococci*. The work in this thesis follows the library-dependent approach with CPLOP as its library. CPLOP uses *E. coli*, taking advantage of its good characteristics, such as not normally being pathogenic to humans, present in the feces of almost all host species and with a genotype unique enough to differentiate between host species [12].

When using FIB for MST, it is necessary to understand the concept of bacterial strain. Different research groups have different definitions for strain but generally, a strain of a microbial species can be thought of as a “group“ or “family“ of microbes that shares a common ancestor. In our case, we think of a bacterial strain as a subtype of bacterial species in which all the microbes are closely related in some meaningful way, specifically in our case, through their pyroprints. By using pyro-DBSCAN as our clustering algorithm with a defined similarity metric and a threshold, the clusters resulted from pyro-DBSCAN can be thought of as bacterial strains in further studies. Another concept worth mentioning is the concept of a transient strain. Strains are considered “transient“ if they contain isolates that come from multiple host species. In other words, transient strains are groups of isolates that can not be assigned to a specific source category.

2.1.2 Pyroprinting

Pyroprinting is a novel DNA sequencing method developed at Cal Poly by Black et al. [4]. Pyroprinting was developed to address the need for a rapid, reproducible, and inexpensive method for bacterial strain typing. It is based on a DNA sequencing method called *pyrosequencing*.

Pyrosequencing is a technique performed by a “pyrosequencer”. The process involves copying only short sequences of DNA, one base at a time, and detecting the amount of light emitted every time when a nucleotide binds to the DNA. In order for the pyrosequencing to work, the researchers need to provide the DNA along with a primer to bind to a specific part of the DNA. A primer is a short strand of RNA or DNA that serves as a starting point for DNA synthesis. Once the primers are added, the DNA is copied in stages called dispensations. Only one type of nucleotides (A, T, C or G) is added in one dispensation. The amount of light emitted in one dispensation is directly proportional to the number of nucleotides that get added to the DNA.

The resulting graph of a pyrosequencing is called *pyrogram*, which includes several peaks. Each peak represents one dispensation (see Figure 2.1). A peak has height, width, and area. Peak height represents the maximum value of the emittance graph for a dispensation. Peak width represents the width of the emittance graph for a dispensation. Peak area represents the area under the emittance graph of a dispensation.

2.1.3 Internal Transcribed Spacers (ITS)

To effectively compare the results of pyrosequencing between isolates, the same region of DNA needs to be sequenced every time. Picking an appropriate region of DNA

Figure 2.1: A sample pyrogram - the result of pyrosequencing. The values represent the heights of the peaks

to sequence is a crucial first step to obtain a good DNA fingerprint. Generally, researchers want to avoid highly conserved regions of DNA. These regions are almost the same in most strains due to its vitality to the survival the organisms. Hence, highly conserved regions are not good representations for strain differentiation. For this reason, the ribosomal RNA genes' *Internal Transcribed Spacers (ITS)* become good candidates. ITS refer to the regions of non-coding DNA located between two highly conserved rRNA genes. In our study, our main focus is the two ITS regions between the 16S and 23S (ITS-1) genes and between the 23S and 5S (ITS-2) genes (see Figure 2.2).

In any living organism, ribosomal RNAs are essential for protein synthesis. The number of copies of rRNA operons per bacterial genome varies from one to as many as 15 [11]. It is a general assumption that this high number of copies of rRNA operons is to ensure the high growth rates of prokaryotic organisms. In *E. coli*, there are seven different rRNA operons. Choosing *E. coli* as our FIB allows us the opportunity to examine seven highly variable regions for each ITS. In addition, the pyroprinting process for these ITS regions is also straightforward and inexpensive. Researchers can simply use the primers to attach to the highly conserved regions of DNA in front of the ITS regions and apply PCR (Polymerase Chain Reaction) to amplify these ITS regions of interest.

Figure 2.2: A simplified diagram of *E. coli* genome, where there are seven copies of rRNA operon. Each rRNA operon contains the two ITS regions of interest, ITS-1 and ITS-2.

2.1.4 Pyroprint

Since we adopted pyrosequencing a little differently in our study, the pyrograms we obtained are also different from the pyrograms of true DNA sequencing. The biologists at Cal Poly decided to call them *pyroprints*. Pyroprints are different from normal pyrograms in that they cannot be used to determine the specific DNA sequence of the region. Pyroprints are obtained from amplifying and sequencing seven variations (loci) of a DNA region of *E. coli*. It is essentially impossible to determine the distribution of added nucleotides among each loci. Therefore, they are considered as DNA fingerprints, representing the unique identifier for that region of a species that can later be used for strain discrimination.

As previously mentioned, a pyrogram consists of peaks and information about each peak's height, width, and area. In order to digitize a pyroprint, researchers can use any of this information, however, previous study done by Shealy indicated that peak heights are more accurate representations compared to peak areas and widths

[13]. These values are represented in a vector form of size D :

$$\vec{p} = (h_1, h_2, \dots, h_D)$$

where D is the number of dispensations of the pyroprint and h_i is the peak height of the i^{th} dispensation.

In order to increase the ability to discriminate between species, for each sample, we perform pyroprinting on two ITS regions: ITS-1 and ITS-2. This gives us two different pyroprint vectors for each isolate.

Since the two ITS regions represent different DNA segments in *E. coli*, when comparing two isolates, we must consider each region separately. After we have the similarity metric of each region, then we can combine their results to make a meaningful decision. The next section details the similarity metric we apply on pyroprints to make sense of the information that they carry.

2.1.5 Pearson Correlation

The *Pearson Correlation coefficient* measures the correlation between two variables. In our case, we use Pearson correlation to compare pyroprints and determine their similarity.

Given two D -dimensional vectors, $\vec{X} = (x_1, \dots, x_D)$ and $\vec{Y} = (y_1, \dots, y_D)$, the Pearson correlation coefficient ρ between $\vec{X}$ and $\vec{Y}$ is defined as follows [2]:

$$\rho(\vec{X}, \vec{Y}) = \frac{\text{cov}(\vec{X}, \vec{Y})}{\sigma_{\vec{X}} \cdot \sigma_{\vec{Y}}} \quad (2.1)$$

where $\text{cov}(\vec{X}, \vec{Y})$ is the covariance between $\vec{X}$ and $\vec{Y}$, $\sigma_{\vec{X}}$ and $\sigma_{\vec{Y}}$ are the standard deviations of $\vec{X}$ and $\vec{Y}$, respectively.

The *covariance* measures the joint variability of two random variables. In simple term, it means how much two variables vary together. A large value of covariance

can mean a strong relationship between variables, while the sign of covariance shows the tendency in their linear relationship. The covariance between $\vec{X}$ and $\vec{Y}$ is defined as:

$$cov(\vec{X}, \vec{Y}) = \frac{1}{D} \sum_{i=1}^D (x_i - \mu_{\vec{X}})(y_i - \mu_{\vec{Y}}) \quad (2.2)$$

where $\mu_{\vec{X}}$, $\mu_{\vec{Y}}$ is the mean values of $\vec{X}$ and $\vec{Y}$, respectively.

The *standard deviation* measures the amount of variation of a set of values. A low standard deviation indicates that the data points are closer to the mean, while a high standard deviation means the data points are more dispersed. For $\vec{X}$, its standard deviation is defined as:

$$\sigma_{\vec{X}} = \sqrt{\frac{1}{D} \sum_{i=1}^D (x_i - \mu_{\vec{X}})^2} \quad (2.3)$$

We can transform the standard deviation in term of covariance as:

$$\sigma_{\vec{X}} = \sqrt{cov(\vec{X}, \vec{X})} \quad (2.4)$$

Replacing equation (2.4) into equation (2.1) gives us the Pearson correlation coefficient for when the two vectors are the same.

$$\rho(\vec{X}, \vec{X}) = \frac{cov(\vec{X}, \vec{X})}{\sigma_{\vec{X}} \cdot \sigma_{\vec{X}}} = \frac{cov(\vec{X}, \vec{X})}{\sqrt{cov(\vec{X}, \vec{X})} \cdot \sqrt{cov(\vec{X}, \vec{X})}} = 1 \quad (2.5)$$

Due to the normalizing effect of the σ values, $\sigma \leq 1$, Pearson Correlation values are also less than or equal to 1, $\rho \leq 1$. In fact, Pearson correlation returns values between -1 and 1, where 1 indicates a perfect match, -1 indicates the diametric opposite of a perfect match, and 0 means the pair is unrelated. In our study, however, because the pyroprints are non-negative, their Pearson correlation coefficients are always greater

than or equal to 0 ($\rho \geq 0$). A study done by Shealy further shows that multiple pyroprints of the same isolate yield a $\rho \geq 0.995$ [13]. We later use this value as a threshold value for quality control and strain evaluation.

2.2 Related Work

2.2.1 Cal Poly Library of Pyroprints - CPLOP

The digital library used for Library Dependent MST method at Cal Poly is called Cal Poly Library of Pyroprints (CPLOP). CPLOP was started in 2011 as a quarter-long class project. At the end of 2011, CPLOP version 1.0 was released with some basic functionalities of storing and managing pyroprints. In 2013, CPLOP version 2.0 was developed and released by Soliman with extensive upgrades [14].

While the physical samples are frozen and stored in a cold storage, their virtual information is stored and maintained in CPLOP. The data structure in CPLOP allows access to all information about individual isolates, from their pyroprints for different ITS regions to their meta-data about sample's name, origin, ID, host and host species. Currently, CPLOP contains over 19,000 pyroprints from *E. coli* isolates that have been collected, isolated and pyroprinted by students and researchers at the Cal Poly Biology department. These pyroprints make up over 8,100 isolates from 83 different host species. A large portion of these isolates was derived from Cow and Human sources. Table 2.2.1 shows the number of isolates for the top 10 host species in CPLOP.

As previously mentioned, CPLOP version 1.0 was very basic. It was released with a web interface that allowed users to upload samples and browse sample data. As a database, CPLOP 1.0 had functionality for creating, reading, updating and deleting data. However, as the demand for a good MST grew, CPLOP version 1.0 quickly

Table 2.1: The number of isolates per host species for the top 10 host species in CPLOP

No.	Host Species	Iso. Count
1	Cow	1749
2	Human	1636
3	Pigeon	233
4	Ground Squirrel	196
5	Wild Turkey	191
6	Dog	179
7	Sheep	94
8	Seagull	78
9	Pig	66
10	Horse	52

demonstrated its limitations, such as the inability to compare pyroprints, the inability to perform strain matching and the lack of data validation techniques upon erroneous submissions. For that, CPLOP 1.0 required more extensive work to expand on its core capabilities.

In 2013, Soliman added new features and released CPLOP version 2.0 [14]. This updated version not only addressed all the limitations in version 1.0, but also improved upon the database storage and its performance. With the new notion of strains added, CPLOP 2.0 can now perform strain matching. The results can be saved and viewed on a web page. An automated system was also added to flag possible erroneous submissions instead of allowing the submission to go through several quality control procedures. The improvements in CPLOP 2.0 allows researchers to do more thorough MST investigations in the future.

2.2.2 Ontology-based Hierarchical Clustering - OHClust!

After establishing a library to store the DNA fingerprints of strains of *E. coli*, researchers needed some methods to help identify strains. OHClust! was the first clustering algorithm specifically designed and developed for CPLOP to accomplish that goal.

OHClust! was developed by Montana. The acronym is short for Ontology-based Hierarchical Clustering [9]. This algorithm is a modification of the agglomerative hierarchical clustering algorithm. Agglomerative clustering utilizes the bottom-up approach, meaning it builds the dendrogram (tree) from the bottom level up (see Figure 2.3). The clustering process starts with single elements and merges them into pairs of clusters using the information from their metadata. The process stops when all the data points are merged into a single cluster. Compared to its counterpart, divisive clustering, agglomerative clustering is more popular, faster, and easier to implement.

To measure the distance between two clusters, OHClust! uses the average-link method. In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters [9]. The algorithm for agglomerative clustering requires $O(N^2)$ steps. Coupling the calculations for average-link, the overall complexity of OHClust! is bounded at $O(N^3)$ [9].

Even though the results of OHClust! were considered acceptable by the biologists, there was no validation method to determine the correctness of strain discrimination produced by OHClust!. Furthermore, when incorporated to CPLOP 2.0, OHClust! took anywhere between several minutes to several hours based on the size of the database [14]. OHClust! also required more RAM than the CPLOP's server can provide. Motivated by the desire to improve the performance of OHClust!, an alternative clustering algorithm called pyro-DBSCAN was developed for CPLOP.

Figure 2.3: A simple dendrogram (tree) representing clusters for a dataset of 5 data points. The process of merging starts at the bottom nodes. From the original 5 nodes 1, 2, 3, 4, and 5, three clusters were formed $(1,2) = (6)$, (3) and $(4,5) = (7)$. Then the clusters (6) combined with (3) to form (8) . Finally, cluster (8) merged with (7) to form cluster (9) .

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

DBSCAN is short for Density-Based Spatial Clustering of Applications with Noise. DBSCAN uses the density-based notion of clusters. The algorithm takes two parameters ε and `minPts` as input and uses them to classify every data point into clusters. While the number of `minPts` determines the minimum number of neighbors a point must have, the value of ε determines the radius of the sphere surrounding a point. This means that the higher the value of `minPts`, the more neighbors a point must have within an ε -distance of it and its neighbors to become a cluster.

In DBSCAN, there are three classifications for a data point: a core point, a border point, and noise. These points are defined as the following [7]:

- **a Core point:** a point with at least `minPts` neighbors.
- **a Border point:** a point with less than `minPts` neighbors but within ε -neighborhood of a core point.
- **Noise:** any point that is not a core point or a border point.

A DBSCAN cluster is defined as a group of core points and the border points that neighbor those core points [7]. Figure 2.4 is an example of a density-based cluster where `minPts` = 3.

In his work, Johnson modified the original DBSCAN and incorporated his modified algorithm to CPLOP to find nearby isolates and identify strains [7]. Johnson’s work in pyro-DBSCAN allows the comparison of isolates on two different ITS regions to determine their similarity metrics. In addition, Johnson also designed and implemented a data structure called “spatial index”, that was tailored specifically for CPLOP to allow a quick search for neighboring points in high-dimensional space.

Figure 2.4: An example of a density-based cluster where $\text{minPts} = 3$.

Starting at the point in the center, it is a core point since it has at least 3 neighboring points. These neighboring points are also core points since they also have at least 3 neighboring points. Expanding from these core points, the points in green are within ε -neighborhood of core points but they do not have at least 3 neighboring points. They are then classified as border points. Some of the points in grey are within ε -neighborhood of border points but not core points. They also do not have enough neighboring points, hence, they are classified as noise.

After running pyro-DBSCAN, we obtain clusters that contain data points that can be categorized into four groups [8]:

- **Hits:** number of isolates from plurality species in impure clusters.
- **Misses:** number of isolates from minority species in impure clusters.
- **Pure points:** number of isolates in 100% pure clusters.
- **Noise:** number of isolates that did not get clustered.

Incorporation of pyro-DBSCAN into CPLOP shows significant improvement in clustering. The accuracy was quite high. An evaluation on pyro-DBSCAN showed that with `minPts` ranging from 1 to 9, at least 40% the isolates get clustered. Among those that get clustered, the accuracy level is above 80% [8]. Johnson’s algorithm run-time is also reduced to a few minutes. Its RAM usage is under 60MB for a data size of ~ 6000 isolates [7]. One disadvantage of pyro-DBSCAN, however, is that even though the accuracy level was high, many isolates were still not clustered and ended up being categorized as noise. Given an unknown isolate, if it falls into this category, then we have no way to determine its host species. Therefore, in addition to clustering, we need another technique to make the MST process more efficient.

2.2.4 k-Nearest Neighbors Resolution Algorithms for Pyroprints - k-RAP

Even though clustering using pyro-DBSCAN achieves relatively high accuracy level, it only works on the data points that do get clustered. The majority of the points fail to be clustered, hence, classified as “*noise*”. In 2016, k-RAP was developed with the primary goal of improving the accuracy level of the MST process. k-RAP is short for k-nearest neighbors Resolution Algorithms for Pyroprints [8]. The reason for these resolution algorithms was due to the fact that each isolate has two different

pyroprints for two ITS regions (ITS-1 and ITS-2). The two pyroprints produce two Pearson correlation values - $\rho_{\text{ITS-1}}$ and $\rho_{\text{ITS-2}}$. When performing the regular k-NN algorithm on an isolate, we produce two different k-NN lists. The methods proposed by k-RAP offer ways to resolve these multiple k-NN lists to determine the isolate's host species.

The *k-Nearest Neighbors* algorithm (k-NN), which k-RAP built on, is a simple and straightforward algorithm. The basic idea of k-NN is that a certain data point is more likely to be similar to its surrounding neighbors than to other points in the dataset. Going off of this assumption, we suppose that for an unknown object, if we can establish a list of its k -nearest neighbors, then we can determine the classification of our unknown object with a higher level of certainty by assigning the most dominant classification of the k-NN list to it.

k-RAP offers four resolution algorithms that are termed and defined as follows [8]:

- **Meanwise Resolution:** Takes the average of the comparison metric values to form a single k-NN list and finds the plural species of that list.
- **Resolution by Winner:** Finds the most plural species of each k-NN list and picks the classification with the higher number of instances.
- **Resolution by Union:** Combines all k-NN lists to form a union set and picks the classification with the highest number of instances
- **Resolution by Intersection:** Finds a set of size k that contains isolates that are in all k-NN lists and picks the plural classification of the resulting set.

When McGovern ran k-RAP on the entire CPLOP database, he obtained the classification accuracy above 50% with most of the resolution algorithms [8]. k-RAP

behavior changed when he varied the values of k neighbors and filtering threshold, α . The α threshold allows researchers to filter out the neighboring isolates that are too dissimilar to compare. Previous studies were done to determine this threshold as well as how to convert it into an appropriate value to use in our comparison [7]. Evaluation done by McGovern showed that incorporating α threshold into k-RAP caused some resolution to perform better than others. According to McGovern, the descending ranking order of these resolutions without α -filtering was: union, intersection, meanwise, and winner. When α -filtering was applied, this order became: intersection, union, meanwise, and winner. The work in this thesis utilizes all four resolution algorithms to create a comprehensive solution in an attempt to improve on the accuracy level of the MST process.

Chapter 3

DESIGN

This chapter describes the design of the algorithm developed for this thesis. It also includes some details and modifications made on related algorithms to fit into the work of this thesis.

3.1 Motivation and Goal

Chapter 2 described the biological background as well as some previous work done at Cal Poly in developing a new Microbial Source Tracking method. This method follows a library-dependent approach, using CPLOP as its main tool. The work in this thesis is also designed specifically for CPLOP. Our ultimate goal is to provide a final solution for MST that combines clustering and k-RAP’s resolution techniques.

Even though clustering and k-RAP achieve an acceptable level of accuracy individually, they still leave some data with no MST label. By combining them, our goal is to improve the accuracy level even further and make sure that all isolates get classified. In the worst case where our solution cannot beat the previous solutions, we still anticipate our solution for the new MST method to be better than most of the current MST methods.

3.2 Methodology

3.2.1 Comparing Isolates

In order to increase the ability to discriminate between strains, researchers at Cal Poly perform pyroprinting on two separate ITS regions of *E. coli* [4]. Each *E. coli*

isolate in CPLOP can be represented as:

$$I = (\vec{p}_1, \vec{p}_2)$$

where $\vec{p}_1$ and $\vec{p}_2$ are the pyroprint vectors of isolate I for the ITS-1 and ITS-2 regions, respectively.

Since each *E. coli* isolate in CPLOP has two pyroprints for two different ITS regions, ITS-1 and ITS-2, in order to compare two isolates effectively, we must compare each of these regions separately and then determine a meaningful method to combine their results. For each region of ITS, we use the Pearson Correlation as our comparison metric. Algorithm 1 describes how we obtain the Pearson Correlation for two isolates at a region of interest, where the formula for Pearson Correlation calculation can be found in Equation 2.1.

Algorithm 1 Pearson Correlation

Input:

- I_1 and I_2 are two isolates in CPLOP
- *region*: ITS region of interest (ITS-1 or ITS-2)

Output:

- $\rho(I_1, I_2)$ indicating their similarity when comparing their pyroprints at the region of interest

```

1: procedure GETPEARSONCORRELATION( $I_1, I_2, region$ )
2:    $\vec{p}_{I_1} \leftarrow \text{GETPYROPRINT}(I_1, region)$ 
3:    $\vec{p}_{I_2} \leftarrow \text{GETPYROPRINT}(I_2, region)$ 
4:   return PEARSONCORRELATION( $\vec{p}_{I_1}, \vec{p}_{I_2}$ )

```

3.2.2 α -filtering

Given an unknown isolate, for each of its ITS regions, we can apply Algorithm 1 to find the similarity metric between it and every existing isolate in CPLOP. From there, we can sort CPLOP isolates in descending order on the similarity and obtain

a list of k-nearest neighbors. A modification in our selection process is an addition of a filtering threshold, called α , which is used by the researchers to filter out isolates that are too dissimilar to compare. The value of α is between 0 and 1. Whenever α is applied, we consider only the top k isolates whose similarity is *above* threshold α . Algorithm 2 outlines the steps to find k-Nearest Neighbors with α threshold.

Our justification to pick a value for the α threshold is based on a previous analysis done by Shealy [13]. In her work, Shealy compared multiple pyroprints of the same isolate and performed statistical analysis on them to determine thresholds for each ITS region. Based on Shealy’s analysis, the researchers picked two thresholds for pyroprinting, α and β [13]. For two isolates, if their Pearson correlation is *above* the α threshold 0.995 in *both* ITS regions, they are considered “definitely similar.” If their Pearson correlation is *below* the β threshold 0.99 in *either* ITS regions, they are considered “definitely dissimilar”. Isolates whose Pearson correlation is between the α and β thresholds may or may not be similar [7]. In the scope of this thesis, we only focus on α -filtering, which is the threshold above which the isolates are considered similar. In our evaluation, we experiment with a few values of α : 0.0 (meaning no filtering), 0.98, 0.99 and 0.995.

3.2.3 The Two-step Classification Technique - HAP

In this work, we propose a two-step classification technique as an enhancement to the current MST method used at Cal Poly. We term this technique HAP which is short for Hybrid Algorithm for Pyroprints since it is a combination of clustering and k-Nearest Clusters algorithm. The algorithm for k-Nearest Clusters is a modification from the algorithm for k-nearest neighbors to fit into our study. It also utilizes the resolution methods developed in k-RAP.

The detail of HAP is as follows. As a starting point, we cluster all the existing

Algorithm 2 k -Nearest Neighbors algorithm with α threshold

Input:

- $list$: a list of elements sorted by a similarity metrics
- k : number of desired elements
- α : filtering threshold

Output:

- a list of k -nearest elements

Requires:

- k and α values are predetermined

```
1: procedure GETNEIGHBORS( $list$ )  
2:    $neighbors \leftarrow \emptyset$            /* Make nearest neighbors list.          */  
3:    $i = 0$   
4:   while  $i < k$  do  
5:     if  $list[i].sim > \alpha$  then       /* Filter elements above the threshold */  
6:       add  $list[i]$  to  $neighbors$       /* Add to neighbors.                  */  
7:        $i \leftarrow i + 1$   
8:   return  $neighbors$ 
```

isolates in the library, sorting out the clusters and the outliers. This information is processed and stored in a dataset. Algorithm 3 describes how to obtain this dataset. Next, given an unknown isolate, the first step requires comparing this isolate against all the clusters. An isolate belongs to a cluster if its distance to the cluster’s centroid is within the cluster’s radius. If the unknown isolate belongs to a cluster, the classification of its host species is returned as the dominant species of the cluster it belongs. We break ties by selecting the cluster where the distance from the cluster’s isolate to the unknown isolate is the smallest. In the case where the unknown isolate does not belong to any cluster, the second step is to apply k-Nearest Clusters algorithm to determine the final classification of the unknown sample. If there is more than one classification, we randomly pick the first one. Algorithm 4 details this process in pseudo-code.

The detailed algorithm and pseudo-code for each of the k-RAP classification methods can be found in [8]. These methods take two parameters: a library of known isolates and an unknown isolate. Their output is the classification of the unknown isolates. In order to use k-RAP in our classifier, we need to convert clusters into “*isolates*” and modify k-RAP to treat these “*isolates*” like normal isolates in the library. To differentiate these from regular isolates in the library, we set up their names to start with the word “*cluster*” and followed by a numeric value. Henceforth, we shall refer to these “*isolates*” as *cluster-isolates*. The “pyroprints” of a cluster-isolate are calculated by taking the arithmetic mean of all the pyroprints of isolates in that cluster. A cluster-isolate’s main species, however, is determined by four different methods:

- **Unweighted Dominant Species selection (DS) method:** The main species of each cluster is its dominant species.
- **Unweighted Collective Species selection (CS) method:** The main species

Algorithm 3 Get ClusterDataset from pyro-DBSCAN

Input:

- $\mathbb{L}$: library of known isolates

Output:

- *dataset*: containing information on clusters and outliers

```
1: procedure GETCLUSTERDATASET( $\mathbb{L}$ )
2:   dataset  $\leftarrow \emptyset$ 
3:   clusters  $\leftarrow$  PYRO-DBSCAN( $\mathbb{L}$ )
4:
5:   /* Add all clusters (“strains”) to dataset */
6:   add clusters to dataset
7:
8:   mod_isolates  $\leftarrow \emptyset$ 
9:   /* Transform clusters into “cluster-isolates” and add them to mod_isolates */
10:  for cluster in clusters do
11:    convert cluster to new_isolate
12:    add new_isolate to mod_isolates
13:
14:  /* Add all outliers (non-clustered isolates) to mod_isolates */
15:  for isolate in  $\mathbb{L}$  do
16:    flag  $\leftarrow$  False
17:    for cluster in clusters do
18:      if isolate in cluster then
19:        flag  $\leftarrow$  True
20:    if flag = False then
21:      add isolate to mod_isolates
22:  add mod_clusters to dataset
23:
24:  return dataset
```

Algorithm 4 2-step Classifier - HAP

Input:

- *unknown*: an unknown isolate
- *dataset*: contains information resulted from clustering
- *resolution*: a k-RAP resolution whether it is meanwise, union, winner or intersection
- *method*: an integer value from 1 to 4, representing one of the four HAP method

Output:

- *s*: a classification for the unknown isolate

Requires:

- *k*, α , ϵ are predetermined

```
1: procedure CLASSIFIER(unknown, dataset, resolution, method)
2:   /* Step 1: Check unknown isolate against each cluster in clusters */
3:   clusterList  $\leftarrow \emptyset$ 
4:   for cluster in dataset.clusters do
5:     dist  $\leftarrow$  GETDISTANCE(unknown, cluster)
6:     if dist  $\leq$  cluster.radius then
7:       add cluster to clusterList
8:
9:   if clusterList not  $\emptyset$  then
10:    /* Check neighboring isolates */
11:    filtered  $\leftarrow \emptyset$ 
12:    for cluster in clusterList do
13:      neighbors  $\leftarrow \emptyset$ 
14:      for isolate in cluster do
15:        if isWithinRadiiOf(unknown, isolate,  $\epsilon$ ) then
16:          add isolate to neighbors
17:        if |neighbors|  $\geq 1$  then
18:          add cluster to filtered
```

```

19:      /* Select the cluster with the smallest distance */
20:      if filtered not  $\emptyset$  then
21:          resCluster  $\leftarrow$  GETCLUSTERWITHSMALLESTDISTANCE(filtered)
22:          s  $\leftarrow$  FINDMOSTPLURALSPECIES(resCluster)
23:          return s
24:
25:  /* Step 2: Perform k-RAP if unknown does not belong to any cluster */
26:  /* Get an appropriate function to find the most plural species */
27:   $\mathbb{F} \leftarrow$  GET_FUNC_TO_FIND_PLURAL_SPECIES(resolution, method)
28:  kRAPClassifier  $\leftarrow$  GETKRAPCLASSIFIER(resolution)
29:  weight  $\leftarrow$  GETWEIGHTINGScheme(method)
30:  s  $\leftarrow$  kRAPClassifier(unknown, dataset.mod_isolates, weight,  $\mathbb{F}$ )
31:  return s

```

of each cluster is a collection/group of all species appear in that cluster.

- **Weighted Dominant Species selection (wDS) method:** Same as DS method with the addition of a weight coefficient.
- **Weighted Collective Species selection (wCS) method:** Same as CS method with the addition of a weight coefficient.

The weighting scheme for wDS and wCS methods is described in detail in Section 3.4.1

3.3 Clustering Data

In order to incorporate the results from pyro-DBSCAN into our algorithm, we need some structure to sort and store the data produced by pyro-DBSCAN. Since the clusters will not change unless the library changes, in order to improve the overall efficiency, we only need to perform clustering with pyro-DBSCAN once and re-run it if there is any change to the library.

3.3.1 Cluster

For each cluster obtained from pyro-DBSCAN, we create a structure to store: all the isolates in the cluster, the size of the cluster, its centroids (one centroid for each ITS region), and its radii (one radius for each ITS region).

A centroid of an object is defined, mathematically, as a point whose position is at the mean position of all the points in all of the coordinate directions. For example, the centroid of a set of k points: $x_1, x_2, \dots, x_k$ is calculated as:

$$C = \frac{x_1 + x_2 + \dots + x_k}{k}$$

Once we have the centroid of a cluster, a cluster’s radius is defined as the distance from the centroid to the furthest isolate.

3.3.2 ClusterDataset

The result from clustering using pyro-DBSCAN is stored in a dataset. This dataset contains information on the clusters and the outliers produced by pyro-DBSCAN. In addition, we also process the clusters and “transform” them into *cluster-isolates*, whose names start with *cluster* and end with a numerical value, for example: “cluster_1”, “cluster_2”. The “pyroprints” of a *cluster-isolate* are the “pyroprints” of its centroids.

3.4 k-Nearest Clusters

The algorithm to perform k-Nearest Clusters is a modification of k-RAP. Its detail can be found in Chapter 5 of McGovern’s work [8]. Since the dataset resulted from pyro-DBSCAN is a mixture of clusters and non-clustered isolates, when k-nearest neighbors algorithm is performed on such dataset, we can consider the non-clustered isolates as a cluster of size one. Hence, we call this algorithm k-Nearest Clusters.

Figure 3.1 uses a simple example to demonstrate how k-Nearest Clusters works and our motivation for developing four HAP methods DS, CS, wDS and wCS. The weighting scheme that we use is described in more detail in Section 3.4.1.

Besides the four HAP methods, we also utilize the four resolution algorithms proposed by k-RAP. The reason for using these resolutions is due to isolates in CPLOP having two entirely separate metrics - two different pyroprint vectors for the ITS-1 and ITS-2 regions. k-Nearest Clusters Algorithm is performed after pyro-DBSCAN. Even though, in pyro-DBSCAN, Johnson already resolved and combined these two metrics to determine strains, if we want to perform k-Nearest Clusters on the collection of clusters and outliers/noise produced by pyro-DBSCAN, then we still have to do the comparison based on the two ITS regions. In his study, McGovern also established that simply combining these two metrics into a single vector and performing Euclidean distance may not be appropriate, while these metrics were perfectly good to be on their own. For that reason, when using k-Nearest Neighbors or Clusters on our dataset, we obtain two separate k-nearest neighbors lists for the ITS-1 and ITS-2 regions. These lists are merged using the resolutions proposed by k-RAP. Figure 3.2 presents some isolates in CPLOP in a two-dimensional space where the dimensions are the ITS-1 and ITS2. In the context of our study, we can also consider these points as cluster centroids and/or outliers produced by pyro-DBSCAN.

3.4.1 Weighting Scheme

Among all four of HAP methods, the first two do not involve the use of a coefficient. For these methods, we make the assumption that every neighbor/isolate's classification contributes equally to the classification of the unknown sample. In code, we can think of their coefficient as being equal to one.

A common weighting scheme of k-nearest neighbors, which is based on linear

Figure 3.1: A simple example describing how the k-Nearest Clusters Algorithm works when $k = 3$. Given a dataset consisting of 3 clusters, 2 outliers and an unknown point. In order to determine the classification of the unknown point, we look at its three nearest neighboring clusters or points. Assuming no weighting scheme is applied and the classification of each cluster is its dominant label, then the unknown point has two neighbors labeled A and one neighbor labeled B, hence, its classification is A. However, the cluster labeled B contains more points than A, so it is not fair to treat it equally as other clusters. Also, if we look at the composition of each of the unknown point's neighbors, the unknown point is neighbored with a total of 11 points, of which, 4 points labeled A and 7 points labeled B. This means the classification of the unknown can also be B. For this reason, we propose different strategies for our k-Nearest Clusters by varying the weight coefficient and the clusters' composition. We also evaluate each strategy to determine which one works best.

Figure 3.2: A simple graph annotating isolates in CPLOP in two-dimensional space where the dimensions are the ITS-1 and ITS-2 regions. As previously mentioned, each isolate in CPLOP can be represented as: $I = (\vec{p}_1, \vec{p}_2)$ where $\vec{p}_1$ and $\vec{p}_2$ are the pyroprint vectors of isolate I for the ITS-1 and ITS-2 regions, respectively. From this graph, we can see that when $k = 3$, for the unknown isolate, its three nearest neighbors for the ITS-1 region are $[4,1,7]$ and for the ITS-2 region are $[3,2,1]$.

interpolation, gives each neighbor a weight of $\frac{1}{d}$, where d is the distance from that neighbor to the point of interest. The justification for this weight assignment is simple: the nearer neighbors should contribute more weight to the unknown sample than the ones farther away. However, since our study involves both clusters and non-clustered isolates, and since the results of our clustering algorithm have such high level of accuracy, it is safe for us to make another assumption that clusters should carry more weight than non-clustered isolates. In addition to the common weighting scheme, our next step is to determine a good coefficient to reflect the contribution of the clusters.

Each cluster, in addition, to the isolates that make up the cluster, has a centroid and a radius for each ITS region. More details on cluster's composition and how to calculate them can be found in Section 3.3.1. It is intuitive to think that the larger a cluster is, the more it should contribute to the unknown isolate's classification. The general idea is that given a cluster and an unknown isolate, we want to maximize the radius of a cluster while minimizing the distance between a cluster and the unknown isolate. For that reason, we decide to pick a coefficient for our clusters such as:

$$C_c = \frac{r}{d(u, cluster)}$$

where C_c is the cluster's coefficient, r is the radius of the cluster of interest and d is the distance between the cluster and the unknown isolate

And for non-clustered isolates, their coefficient is calculated as:

$$C_i = \frac{1}{d(u, isolate)}$$

where C_i is the isolate's coefficient and d is the distance between the isolate of interest and the unknown isolate. A simple example for this is described in Table 3.1

Depending on which HAP method being selected, an appropriate main species will be assigned to each cluster. Then depending on a resolution method, we can determine the classification of our unknown sample. The classification step is where

Table 3.1: A simple example describing our weighting scheme. Even though cluster-1 is further from the unknown than isolate-1, cluster-1 has a radius of 2, meaning the furthest point from the cluster is only 0.5 apart from the unknown. That means the classification of cluster-1 should be weighted more than isolate-1. Given our weighting scheme, we can calculate the classification for the unknown isolate as such: Human = $1/1.5 = 0.67$; Cat = $2/2.5 = 0.8$. Therefore, the final classification for the unknown isolate is “Cat.”

Isolate	Radius	Distance to unknown	Host Species	Weight
Unknown Isolate	-	0	?	-
Isolate-1	-	1.5	Human	0.67
Cluster-1	2	2.5	Cat	0.8

a weighting scheme is applied. The following subsections detail the modifications made on each of the resolution methods in k-RAP.

3.4.2 Modifications to Meanwise Resolution

Given an unknown isolate, Meanwise Resolution builds a single k-nearest neighbors list using the arithmetic means of all the comparison metrics between the unknown isolate and each of the isolates in the library. The final classification is the most dominant host species in that list. The arithmetic mean is simply calculated by adding all the similarities (one for each ITS region) and dividing by the number of comparison metrics.

For the α filtering, McGovern used the Euclidean norm mapping $([0,1], \dots, [0,1]) \rightarrow ([0, \sqrt{1^2 + \dots + 1^2}])$ to convert the α value to a new threshold. However, by multiplying the α value by a factor of $\sqrt{2}$, for any $\alpha > 0.71$, it inadvertently filters all isolates because the Pearson Correlation is restricted in the range $[0, 1]$ and $\alpha_{new} > 0.71 * \sqrt{2} > 1.0$. For this reason, McGovern could not use Meanwise with α filtering. In order to use Meanwise with α filtering, we decided to keep the α value as is. Our reasoning is that for two given isolates, in order for them to be considered “similar”,

they need to have the Pearson Correlation of *both* ITS regions greater than α ($\alpha = 0.995$). Even when we take the mean value for the Pearson Correlation of these ITS regions, the mean value still needs to be greater than α , hence there is no reason to change α .

In our classifier, a few modifications are added to the original Meanwise Resolution algorithm. The original algorithm for Meanwise only takes two parameters: the unknown isolate and the library of known isolates. However, since each HAP method has a different way of assigning the clusters' main species, two additional parameters are added: a boolean value to determine if we want to use weight coefficients on the clusters and a function $\mathbb{F}$ to find the most dominant species. Algorithm 5 describes this process in pseudocode. A simple example for Meanwise Resolution is demonstrated in Figure 3.3.

k	Isolate x	Host Species	$\rho_{its-1}(u, x)$	$\rho_{its-2}(u, x)$	$mean(\rho_{its-1}, \rho_{its-2})$
1	a	Cat	0.994	0.991	0.993
2	b	Dog	0.990	0.994	0.992
3	c	Dog	0.995	0.989	0.992
4	d	Chicken	0.985	0.987	0.986
5	e	Dog	0.980	0.990	0.985

(a) In this example, the classification of the unknown isolate is “Dog” since “Dog” has the max count of 3.

No.	Host Species	Count	Weight Coefficient	Final Count
1	Dog	3	0.2	0.6
2	Cat	1	0.6	0.6
3	Pigeon	1	0.8	0.8

(b) Even though “Dog” has the most count originally, when taking the weight coefficients into account, the final classification is “Pigeon”. The final count is the product of the original count and the weight coefficient.

Figure 3.3: A simple example of classifying an unknown isolate u using Meanwise Resolution with $k = 5$.

Algorithm 5 Modified Meanwise Resolution for HAP

Input:

- u : an unknown isolate
- $\mathbb{L}$: a library of known isolates
- $weight$: a boolean value, *True* if we want to apply weight coefficients on the neighbors and *False* otherwise
- $\mathbb{F}$: a function to find the most plural species given a list of isolates

Output:

- s : a classification of the unknown isolate u

Requires:

- k , α , and the set of comparison metrics are predetermined

```
1: procedure MEANWISECLASSIFY( $u, \mathbb{L}, weight, \mathbb{F}$ )
2:    $N \leftarrow \{\emptyset\}$                                 /* Make nearest neighbors list */
3:   for  $p \in \mathbb{L}$  do                                    /* For each isolate in the library */
4:      $sum \leftarrow 0$ 
5:     for  $C_i \in \mathbb{C}$  do                                /* For each comparison metric */
6:        $sim \leftarrow \text{GETPEARSONCORRELATION}(u, p, C_i)$ 
7:        $sum \leftarrow sum + sim$ 
8:      $mean \leftarrow \text{MEAN}(sum)$                         /* Mean the results */
9:     add  $(mean, p)$  to  $N$                             /* Add mean to list of neighbors */
10:  sort  $N$  by  $sim$                                     /* Sort similarities in descending order */
11:   $neighbors \leftarrow \text{GETNEIGHBORS}_{k,\alpha}(N)$  /* Keep the k-nearest neighbors */
12:   $s \leftarrow \mathbb{F}(neighbors, weight)$                 /* Find the most plural species */
13:  return  $s$ 
```

3.4.3 Modifications to Winner Resolution

Winner Resolution determines its classification by building two k-nearest neighbors lists (one for each ITS region), and then picking the most plural species in each list. The final classification is the “winner” of the two lists, meaning the one with the highest instances of the two lists.

k (ITS-1)	Host Species	k (ITS-2)	Host Species
1	Cat	1	Dog
2	Dog	2	Cat
3	Dog	3	Bat
4	Chicken	4	Cat
5	Dog	5	Bat

(a) k-NN list of size 5 for ITS-1

(b) k-NN list of size 5 for ITS-2

Figure 3.4: A simple example of classifying an unknown isolate u using Winner Resolution with $k = 5$. The “winner” species for ITS-1 is “Dog” with the max count of 3. In ITS-2, “Bat” and “Cat” both have the max count of 2. The algorithm randomly picks the first one in the list as its classification, assumingly in this case “Cat.” The overall “winning” species is “Dog” with the max count of 3. When taking into account the weight coefficient, the final counts might be different but the same concept applied.

Similar to Meanwise, the modifications to winner involve the addition of two parameters: a boolean value to determine if we want to use weight coefficients on the clusters and a function $\mathbb{F}$ to find the most plural species. Algorithm 6 describes this process in pseudocode. Figure 3.4 shows how Winner Resolution works.

Algorithm 6 Modified Winner Resolution for HAP

Input:

- u : an unknown isolate
- $\mathbb{L}$: a library of known isolates
- $weight$: a boolean value, *True* if we want to apply weight coefficients on the neighbors and *False* otherwise
- $\mathbb{F}$: a function to find the most plural species given a list of isolates

Output:

- s : a classification of the unknown isolate u

Requires:

- k , α , and the set of comparison metrics are predetermined

```
1: procedure WINNERCLASSIFY( $u, \mathbb{L}, weight, \mathbb{F}$ )
2:    $N \leftarrow \{\emptyset\}$                                 /* New list to track neighbor lists. */
3:   for  $C_i \in \mathbb{C}$  do                                  /* For each comparison metric */
4:      $N_i \leftarrow \emptyset$                           /* Make nearest neighbors list */
5:     for  $p \in \mathbb{L}$  do                                  /* For each isolate in the library */
6:        $sim \leftarrow \text{GETPEARSONCORRELATION}(u, p, C_i)$ 
7:       add  $(sim, p)$  to  $N_i$                           /* Add to neighbors */
8:       sort  $N_i$  by  $sim$                                 /* Sort similarities in descending order */
9:       add  $N_i$  to  $N$ 
10:   $S \leftarrow \{\emptyset\}$                             /* To track each list's most plural. */
11:  for  $N_i \in N$  do                                    /* For each comparison metric */
12:     $neighbors \leftarrow \text{GETNEIGHBORS}_{k,\alpha}(N_i)$  /* Keep the nearest */
13:     $s \leftarrow \mathbb{F}(neighbors, weight)$                 /* Find the most plural species */
14:    add  $s$  to  $S$ 
15:  return  $\text{MAX}(S)$                                      /* The "winning" species overall */
```

3.4.4 Modifications to Union Resolution

Union Resolution determines its classification by building two k-nearest neighbors lists (one for each ITS region), and then finding all the species and their instances

k (ITS-1)	Host Species	k (ITS-2)	Host Species
1	Cat	1	Pigeon
2	Dog	2	Cat
3	Dog	3	Cat
4	Cat	4	Cat
5	Dog	5	Bat

(a) k-NN list of size 5 for ITS-1

(b) k-NN list of size 5 for ITS-2

Host Species	Count
Cat	5
Dog	3
Bat	2
Pigeon	1

(c) The resulting Union set

Figure 3.5: A simple example of classifying an unknown isolate u using Union Resolution with $k = 5$. Even though in ITS-1, “Dog” is the most dominant species. In ITS-2, “Cat” has the maximum count. The resulting union set shows “Cat” as the final classification since it has the highest combined count overall. When taking into account the weight coefficient, the final counts might be different but the same concept applied.

in each list. The final classification is the most dominant species in the resulting “union” of the two lists.

Similarly to previous methods, the modification to Union also involves the addition of two parameters: a boolean value to determine if we want to use weight coefficients on the clusters and a function $\mathbb{H}$ to find all the species. Algorithm 7 describes this process in pseudocode. Figure 3.5 shows how Union Resolution works.

Algorithm 7 Modified Union Resolution for HAP

Input:

- u : an unknown isolate
- $\mathbb{L}$: a library of known isolates
- $weight$: a boolean value, *True* if we want to apply weight coefficients on the neighbors and *False* otherwise
- $\mathbb{H}$: a function to find all the species

Output:

- s : a classification of the unknown isolate u

Requires:

- k , α , and the set of comparison metrics are predetermined

```
1: procedure UNIONCLASSIFY( $u, \mathbb{L}, weight, \mathbb{H}$ )
2:    $N \leftarrow \{\emptyset\}$                                 /* New list to track neighbor lists. */
3:   for  $p \in \mathbb{L}$  do                                    /* For each isolate in the library */
4:     for  $C_i \in \mathbb{C}$  do                                /* For each comparison metric */
5:        $sim \leftarrow \text{GETPEARSONCORRELATION}(u, p, C_i)$ 
6:       add  $(sim, p)$  to  $N_i$                           /* Add to neighbors */
7:       sort  $N_i$  by  $sim$                                 /* Sort similarities in descending order */
8:       add  $N_i$  to  $N$ 
9:    $\mathbb{M} \leftarrow \{\emptyset\}$                             /* A union list of species */
10:  for  $N_i \in N$  do                                    /* For each comparison metric */
11:     $neighbors \leftarrow \text{GETNEIGHBORS}_{k,\alpha}(N_i)$  /* Keep the nearest */
12:     $S \leftarrow \mathbb{H}(neighbors, weight)$                 /* Get list of all species */
13:    add  $S$  to  $\mathbb{M}$ 
14:  return  $\text{MAX}(\mathbb{M})$                                     /* The plural species in union list */
```

3.4.5 Modifications to Intersection Resolution

Intersection Resolution is the most restrictive and complicated resolution in k-RAP. This method determines its classification by building a single set of size k consisting of isolates that appear in both k -nearest neighbors lists (one for each ITS region). In

order to obtain a single list of size k , it requires expanding the two nearest neighbors lists until reaching the desired size. In the case where we cannot expand any further due to α filtering, it may result in a set of size less than k . More details on this algorithm can be found in [8].

The modification to Intersection also involves the addition of two parameters: a boolean value to determine if we want to use weight coefficients on the clusters and a function $\mathbb{F}$ to find the most plural species. Algorithm 8 describes this process in pseudocode. Figure 3.6 shows how Intersection Resolution works.

k (ITS-2)	Isolate ID	Host Species
1	12	Dog
2	4	Dog
3	5	Bat
4	6	Cat
5	1	Dog
6	7	Human
7	10	Cow
8	9	Bat

(a) k-NN list for ITS-1

k (ITS-2)	Isolate ID	Host Species
1	9	Bat
2	6	Cat
3	12	Dog
4	11	Cow
5	13	Human
6	7	Human
7	3	Pigeon
8	1	Dog

(b) k-NN list for ITS-2

k	Isolate ID	Host Species
1	12	Dog
2	6	Cat
3	9	Bat
4	7	Human
5	1	Dog

(c) The resulting Intersection set

Figure 3.6: A simple example of classifying an unknown isolate u using Intersection Resolution with $k = 5$. The k-NN lists of size 5 of both ITS-1 and ITS-2 only yield 3 isolates in common. Since we want an intersection of size 5, we must keep expanding the two k-NN lists. The intersection set is achieved with when we extend the k-NN lists to size 8. The final classification is Dog, with the max count of 2.

Algorithm 8 Modified Intersection Resolution for HAP

Input:

- u : an unknown isolate
- $\mathbb{L}$: a library of known isolates
- $weight$: a boolean value, *True* if we want to apply weight coefficients on the neighbors and *False* otherwise
- $\mathbb{F}$: a function to find the most plural species given a list of isolates

Output:

- s : a classification of the unknown isolate u

Requires:

- k, α, δ and the set of comparison metrics are predetermined

```
1: procedure INTERSECTIONCLASSIFY( $u, \mathbb{L}, weight, \mathbb{F}$ )
2:    $N \leftarrow \{\emptyset\}$                                 /* A list to track neighbors lists */
3:   for  $C_i \in \mathbb{C}$  do                                /* For each comparison metric */
4:      $N_i \leftarrow \emptyset$                         /* Make nearest neighbors list */
5:     for  $p \in \mathbb{L}$  do                                /* For each isolate in the library */
6:        $sim \leftarrow \text{GETPEARSONCORRELATION}(u, p, C_i)$ 
7:       add  $(sim, p)$  to  $N_i$                         /* Add to neighbors */
8:       sort  $N_i$  by  $sim$                             /* Sort similarities in descending order */
9:       add  $N_i$  to  $N$ 
10:   $done \leftarrow \text{false}$ 
11:  while  $\neg done$  do
12:     $N' \leftarrow \{\emptyset\}$                       /* A list to track nearest neighbors lists */
13:    for  $N_i \in N$  do
14:       $N'_i \leftarrow \text{GETNEIGHBORS}_{k,\alpha}(N_i)$ 
15:      add  $N'_i$  to  $N'$ 
16:      if  $|N'_1 \cap \dots \cap N'_n| < k$  then
17:         $k \leftarrow k + \delta$ 
18:      else
19:         $N_\cap \leftarrow N'_1 \cap \dots \cap N'_n$ 
20:         $done \leftarrow \text{true}$ 
21:   $s \leftarrow \mathbb{F}(N_\cap, weight)$                     /* Find the most plural species */
22:  return  $s$ 
```

Chapter 4

IMPLEMENTATION

This chapter describes the implementation of the work contributed with this thesis. It also serves as a reference for any future research. The code, as well as other relevant documentation, can be found at <https://github.com/tramxme/MST-Thesis>.

4.1 Language and Libraries

The previous work done by Johnson on DBSCAN was written in Python while the work done by McGovern on k-RAP was written in Java [7, 8]. Since Python is easy to learn, read, and prototype the solution, we decided to continue using Python for our solution. The code was run with Python 3.6.3 but it should be compatible with any version 3.X of Python.

We continue to make use of many libraries previously used in pyro-DBSCAN such as *numpy*, *mysql*, and *pickle*. *Numpy* is a library for scientific computing with Python. It can perform many sophisticated mathematical functions efficiently and it contains a powerful N-dimensional array object. Since our pyroprints are vectors of ~ 100 dimensions, *numpy* is perfect for storing and comparing pyroprints. *MySQL* is a library that handles all database connection. It is used as a tool to interface with CPLOP. *Pickle* is used for serializing and de-serializing a Python object structure. We mostly used this library for caching while testing - to reduce the need to re-run part of the code. In addition to the aforementioned libraries, we also included other libraries for evaluation purposes, such as *xlwt* for writing information to excel and *pyplot* for graphing the results.

4.2 Overview

4.2.1 k-RAP

The original work of k-RAP was done by McGovern in Java [8]. In order to incorporate k-RAP into our implementation, some modifications need to be made. Since we pick Python as our language of choice, rewriting the k-RAP code in Python is the first important step. Once the code is in Python, modifications are made to it to perform k-Nearest Clusters.

There are four modules for k-RAP: `meanwise`, `union`, `winner`, and `intersection`. These modules contain classes for each resolution method. Each class initialization requires an unknown isolate, a library of known isolates, a list of ITS regions, a mapping to go from isolate's name to species name, a boolean value for weighting scheme and a function to determine the dominant species. Each class contains a method `classify()` that takes in a value of k and α to perform the classification process. Each of the four resolutions returns the predicted host species for the unknown isolate.

The `kNNEval` module is optional. It provides functions that perform evaluation on results returned by the k-Nearest Neighbors algorithms. The pseudocode for these functions can be found in Chapter 3.

4.2.2 k-Nearest Clusters

The `classifier` module contains the classifier, which is also the main contribution of this thesis. Initialization of a classifier requires a configuration, an integer from 1 to 4 representing a HAP method, the unknown isolates, a library of isolates and a mapping to go from isolate's name to species name. The classifier class composes of two main methods: `clustering_classify()` and `kNC_classify()`. The pseudocode for the classifier is described in Algorithm 4. The method `clustering_classify()`

first obtains the dataset resulted from pyro-DBSCAN, and it then checks the unknown isolate against the resulting clusters. An isolate belongs to a cluster if the distance between the isolate and the cluster's centroid is within the cluster's radius. We break ties by selecting cluster whose distance to the isolate is the smallest. If the unknown isolate falls into a cluster, then returns the cluster's dominant species as the classification of the unknown, otherwise, returns None. The second step of the algorithm is performed by method `kNC_classify()`. This method accepts a resolution, a value of k , and a value of α . Depending on which resolution is selected, a k-RAP resolution class is initialized to perform the k-Nearest Clusters algorithm. Method `kNC_classify()` returns the predicted host species for the unknown isolate.

The `kNCEval` module contains functions that perform evaluation on results returned by the HAP classifier. The following chapter describes the evaluation process in more detail.

Chapter 5

EVALUATION

In order to know how well our classifier performs compared to its predecessors, we need to evaluate its accuracy level and compare the results with those of other work. In addition, we also want to focus on answering the following:

- How good is our clustering method in terms of *entropy*?
- What is the value of `minPts` for pyro-DBSCAN that achieves the best result?
- What is the value of `k` for the k-Nearest Clusters algorithm to achieve the best result?
- How effective is our modified k-RAP compared to McGovern's?
- How effective is our HAP classifier compared to the individual performance of pyro-DBSCAN and k-RAP?

Thus far, evaluation on clustering using pyro-DBSCAN and k-RAP shows that when performing pyro-DBSCAN on CPLOP with `minPts` ranging from 1 to 9, more than 40% of the isolates get clustered. Out of the isolates that get clustered, the percentage of correctly categorized isolates is more than 80%. When k-RAP is performed on the entire CPLOP database, the accuracy level is above 50% with most of the resolution algorithms. Based on this initial data, we explore even further to not only see how well our clustering using pyro-DBSCAN works, but also how effective our classifier is compared to pyro-DBSCAN and k-RAP individually.

5.1 Entropy

Entropy is one of the evaluation measures for clustering. It represents the amount of disorder in a cluster. In simpler term, it shows how pure the bacterial strains created by our clustering algorithm are. For each cluster, we can measure its entropy as follows:

$$Entropy(D_i) = - \sum_{j=1}^k Pr_i(c_j) * \log_2 Pr_i(c_j) \quad (5.1)$$

where $Pr_i(c_j)$ is the proportion of class c_j data points in cluster D_i .

The total weighted entropy considering all clusters is calculated as:

$$Entropy_{total}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} * Entropy(D_i) \quad (5.2)$$

where $|D_i|$ is the total number of data points in cluster D_i , $|D|$ is the total data points, and $Entropy(D_i)$ is the entropy of cluster D_i .

The smallest possible value for entropy is 0 which occurs when all species in a cluster are the same, meaning there is no disorder in the cluster. The closer the value of entropy to 0 means the better the clustering algorithm. Figure 5.1 shows a simple example of how to calculate a cluster's entropy and the total weighted entropy.

Table 5.1 lists the weighted entropy and purity obtained from pyro-DBSCAN where the value of **minPts** varies from 1 to 9. All the weighted entropies obtained from pyro-DBSCAN for each **minPts** are under 1.0. The average value is 0.55, indicating that our clustering algorithm did a good job at discerning amongst bacterial strains. The values of entropy decreases as **minPts** increases. This trend makes sense because the number of clusters also decreases as we increase **minPts** and since entropy tends to increase with the number of clusters, the fewer the number of clusters, the lower

Cluster	Cow	Dog	Human	Total	Entropy
1	31	0	1	32	0.2
2	0	0	5	5	0.0
3	10	1	2	13	0.991
Total	41	1	8	50	0.385

Figure 5.1: A simple example showing how to calculate a cluster’s entropy and the overall entropy. The entropy for the individual cluster can be calculated as showed:

$$Entropy_1 = -\frac{31}{32} * \log_2(\frac{31}{32}) - \frac{1}{32} * \log_2(\frac{1}{32}) = 0.2$$

$$Entropy_2 = -\frac{5}{5} * \log_2(\frac{5}{5}) = 0$$

$$Entropy_3 = -\frac{10}{13} * \log_2(\frac{10}{13}) - \frac{1}{13} * \log_2(\frac{1}{13}) - \frac{2}{13} * \log_2(\frac{2}{13}) = 0.991$$

$$Entropy_{total} = 0.2 * \frac{32}{50} + 0.991 * \frac{13}{50} = 0.385$$

the entropy. Table 5.1 also includes the weighted purity. More detail on clustering purity can be found in [8] and in the Appendix A.

5.2 minPts & k

5.2.1 Narrowing minPts for pyro-DBSCAN

Since clustering purity had been investigated by McGovern in [8], we do not go into details in this study. However, in order to narrow down a value of `minPts` for pyro-DBSCAN that achieves the best clustering result, it is essential to examine both the clustering entropy and purity. The definition and calculation for purity can be found in the appendix A.

In order for a clustering algorithm to be considered effective, it needs to have a low entropy and a high purity. The value for entropy indicates the degree of disorder

Table 5.1: The weighted entropy and weighted purity of clusters obtained from pyro-DBSCAN with minPts vary from 1 to 9. We do not want to examine any minPts > 9 because of the large decrease in the number of isolates that get clustered. The detail on how to calculate the weighted purity can be found in the Appendix A.

minPts	1	2	3	4	5	6	7	8	9
# of clusters	387	188	127	88	73	66	59	53	46
% clustered isolates	73.2	64.9	59.8	55.3	52.7	50.7	48.6	46.8	44.8
% correctly clustered isolates (out of total isolates)	62.4	54.4	51.6	47.8	45.6	43.9	42.3	40.9	39.1
% correctly clustered isolates (out of clustered isolates)	85.2	83.8	86.3	86.4	86.4	86.6	87.0	87.3	87.4
weighted Entropy	0.546	0.608	0.556	0.538	0.535	0.514	0.495	0.483	0.479
weighted Purity	0.852	0.838	0.862	0.861	0.863	0.866	0.870	0.873	0.873

in a cluster, while the value for purity indicates its level of accuracy. A perfect cluster has an entropy of 0 and a purity of 1, meaning it contains all the elements of the same class. In our case, a perfect cluster contains all isolates of the same host species.

We generally do not want `minPts` values that are too small because they produce too many clusters of small sizes. However, we also do not want `minPts` values that are too big either because too many isolates end up not getting clustered. In order to maximize the number of isolates that get clustered and also maintain the integrity of the clusters (meaning low entropy and high purity), we decided to narrow down and recommend to use `minPts` values of 3, 4, and 5 for our classifier and for further evaluation process.

5.2.2 Narrowing k for k-Nearest Clusters

Similar to the way `minPts` was used for pyro-DBSCAN, we want to narrow down some values of k that perform best in k-RAP to use in our algorithm. We generally do not want any k values that are too small. A small value for k means that outlier or noise can have a high influence on the final result. Neither do we want a high value of k since it is computationally expensive and it defeats the purpose of the simplicity that k-NN offers.

When McGovern adjusted k values for his study, he found that for $k \geq 5$, the accuracy level does not improve but level off instead [8]. When we re-run k-RAP with the isolates in our database, we also come to similar conclusion. Figure 5.2 demonstrates the overall accuracy level for four different resolution methods when $\alpha = 0.0$. This is performed on our entire database of 4785 isolates. The makeup of the dataset can be found in Table 5.2. In order to maintain a good balance between accuracy and filtering of isolates, we decided to stay with $5 \leq k \leq 9$ when evaluating our classifier.

Figure 5.2: The overall precision of all classifications performed against CPLOP across 4 different resolution algorithms with $\alpha = 0.0$. The accuracy levels off for $k \geq 5$. It reinforces our decision of staying with $5 \leq k \leq 9$.

In addition to using `minPts` and `k` in our evaluation process, we also use an α threshold to filter out dissimilar isolates. More specifically, we examine how our classifier performs when $\alpha = 0.0$ (meaning no filtering), 0.98, 0.99 and 0.995. The next section describes our evaluation process.

5.3 Cross-validation with Holdout

In order to determine the effectiveness of our classifier in classifying the host species of an isolate, we perform cross-validation with holdout against our CPLOP library. The steps for our cross-validation process are as follows:

1. Perform pyro-DBSCAN once on all isolates in CPLOP and store this information to be used throughout the entire process.
2. For each isolate in CPLOP, compare it against the clusters resulting from pyro-DBSCAN. If the distance between an isolate and a cluster's centroid is within the cluster's radius, then that isolate belongs to the cluster and its classification is the dominant species in that cluster. We break ties by selecting the cluster whose distance to the isolate is the smallest.
3. If an isolate does not belong to any cluster, perform k-Nearest Clusters using the four HAP methods, each of which uses four k-RAP resolutions. For each pair of decision strategies, we have a classification of the isolate.
4. Verify whether the classification is correct by checking it against CPLOP since each isolate in CPLOP has a correct host species listed.

For steps 1 and 2 above, we understand that the correct technique should be: for each isolate that we test, we need to remove it from CPLOP, perform clustering, and then compare the isolate of interest against the resulting clusters. However, it is

not efficient to perform clustering every time we test an isolate. Thus, we make an assumption that the addition/removal of one data point from CPLOP will not cause any significant effect on the overall clustering. Based on that assumption, we cluster all the isolates in CPLOP only once and keep using this information throughout our evaluation.

Since the CPLOP library contains many isolates that are not relevant to our study, we set up a few conditions to filter out such isolates and only evaluate our classifier on the dataset that meets our requirements. The first filter is for any isolates that are environmental such as isolates whose common names are “SLO Creek Water”, “unknown Avian”, “Pennington Creek Water”, etc (the full list can be found in `strainpurity` module). We also want to ignore any extremely under-represented isolates because they are unlikely to ever dominate a k-NN list or a cluster. Specifically, in this case, we ignore any isolate whose host species count are less than four.

Table 5.2: The number of isolates per host species filtered from CPLOP that we considered in this study.

Species	No. of Isolates
Cow	1749
Human	1636
Pigeon	233
Ground Squirrel	196
Wild Turkey	191
Dog	179
Sheep	94
Seagull	78
Pig	66
Horse	52

Cat	46
Chicken	44
Bat	37
Mountain Lion	32
Cliff Sparrow	28
Deer	20
White Crowned Sparrow	15
Opossum	12
Sea Otter	10
Pelican	8
Owl	6
Bear	6
California Sea Lion	6
Red Tailed Hawk	5
Common Loon	4
Great Horned Owl	4
Red-shoulder Hawk	4
Bobcat	4
Coyote	4
Grey Fox	4
Elephant Seal	4
Rabbit	4
Raccoon	4
Total	4785

Our final dataset contains 4,785 isolates. From Table 5.2, we can see that the majority of the isolates originate from cows and humans. This certainly skews our classifications for other non-human and non-cow isolates. We discuss more about this in the next section.

In addition to having a good dataset, we also need to have a set of evaluation metrics. The three measures that we will be using in our evaluation are precision, recall, and F-score.

5.3.1 Precision

Precision tracks the accuracy of our classification method. The precision per host species is calculated as:

$$Precision = \frac{|Correct_X \cap Returned_X|}{|Returned_X|} \quad (5.3)$$

where $Correct_X$ is the number of isolates from species “X” in the library, and $Returned_X$ is the number of isolates returned by our classification and labeled as “X.”

For example, if our method returned 100 isolates labeled “Cow” and of which, 75 isolates did come from “Cow,” then our precision is 75%. Since our database contains many species, in this study, we place more focus on the overall precision which is the percentage of isolates that come from any type of species (dog/human/cow/etc) that are correctly categorized by our algorithm. We will also look at the precision of individual host species but only for a few prominent ones.

In our evaluation, we will also be looking at the overall accuracy, which measures the percentage of correctly categorized isolates out of the total isolates. Without filtering, the overall precision is the same as the overall accuracy. These two values differ when α filtering is applied since there will be isolates that will not get classified

and returned as “Undefined.” For example, given a database of 150 isolates, of which 100 isolates came from “Cow,” when we run the classifier and it labels 100 isolates as “Cow” and 50 isolates as “Undefined.” If out of the 100 isolates returned, 75 of them did come from “Cow,” then the precision is 75%. However, the overall accuracy is only at 50%.

5.3.2 Recall

Recall measures how well our classification is at discovering isolates from a given category. The recall for each host species is defined as:

$$Recall = \frac{|Correct_X \cap Returned_X|}{|Returned_X|} \quad (5.4)$$

where $Correct_X$ is the number of isolates from species “X” in the library, and $Returned_X$ is the number of isolates returned by our classification and labeled as “X.”

For example, if our database has 100 isolates that came from “Cow,” our algorithm was able to classify 75 of them as having come from “Cow,” then our recall is 75%. Similar to precision, we put more emphasis on the overall recall and will only demonstrate the individual recall with prominent species.

5.3.3 F-score

F-score is a measure of a test’s accuracy. It takes into account both the precision and recall. The F_1 score is the harmonic average of the precision and recall. It is defined as follows:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (5.5)$$

When the values for precision and recall are 1’s, meaning we obtain perfect pre-

cision and recall, F_1 score reaches its best value at 1. F_1 worst value is 0 when either the precision or the recall is 0. The goal is to maximize F_1 as much as we can, but a value of F_1 above 0.5 means our classification is acceptable.

5.4 Results & Discussion

5.4.1 k-Nearest Neighbors

Since we made some modifications to the resolution algorithms of k-RAP, we performed k-RAP on our dataset and compared our results with that of McGovern in [8]. Figure 5.3 contains the visual graphs of our results, while Table 5.3 shows our results together with those of McGovern.

In his study, McGovern evaluated his k-RAP on a dataset of 4682 isolates with $\alpha = 0.0, 0.98$ and 0.99 . He concluded that with no α filtering, Meanwise Resolution performed third best. Winner Resolution performed worst. Union Resolution performed very well but was not responsive to the change of α filtering. Intersection Resolution performed best when α was applied. He recommended moving forward with Intersection Resolution. The results from our modified k-RAP agreed with some of McGovern’s conclusions while disagreed with others.

Overall for all resolutions, we obtained higher accuracy levels than McGovern. Even when we accounted for isolates that did not get classified, our accuracy stays above 65%. The first explanation for having different results can be attributed to us using different datasets. Our dataset contains 4785 isolates, over 100 isolates more than McGovern’s. Furthermore, our dataset has slightly different makeup than the one McGovern used. The introduction of the new isolates certainly changed the k-NN lists for many isolates in the dataset. The second reason might be due to our way of breaking ties. We are unsure of how the original k-RAP does it since McGovern did

Table 5.3: A comparison of McGovern’s results (colored in green) and our results when performing k-RAP for k ranging from 1 to 10 and $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995 . McGovern’s results are taken from chapter 6 in [8]. For our results, in addition to the overall precision, we also calculate the average accuracy out of total isolates, which includes isolates that do not get classified. An empty cell means there is no available data for that particular pair of resolution and α .

α	0.00	0.95	0.98	0.99	0.995
Meanwise (McGovern’s)	73.20	–	–	–	–
Meanwise	82.73	82.81	83.82	87.06	92.71
Meanwise (Out of total isolates)	82.73	82.78	82.82	81.00	74.57
Winner (McGovern’s)	65.90	–	66.40	68.00	–
Winner	81.80	81.81	81.90	82.25	83.12
Winner (Out of total isolates)	81.80	81.81	81.87	82.10	82.11
Union (McGovern’s)	76.40	–	76.70	76.60	–
Union	82.96	82.96	83.00	83.39	84.19
Union (Out of total isolates)	82.96	82.96	82.96	83.24	83.17
Intersection (McGovern’s)	74.70	–	78.00	85.90	–
Intersection	81.24	81.54	83.00	87.65	93.90
Intersection (Out of total isolates)	81.24	81.30	80.56	77.70	68.74

Table 5.4: The number of isolates that are returned as “Undefined” by our modified k-RAP for $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995 .

α	0.00	0.95	0.98	0.99	0.995
Winner	0	0	2	9	58
Union	0	0	2	9	58
Meanwise	0	2	57	333	936
Intersection	0	14	141	543	1282

(a) $\alpha = 0.95$

(b) $\alpha = 0.98$

(c) $\alpha = 0.99$

(d) $\alpha = 0.995$

Figure 5.3: The overall precision and overall accuracy obtained from our modified k-RAP at $\alpha = 0.95, 0.98, 0.99$ and 0.995 . The overall accuracy values are labeled as “Adjusted.”

not specify how he dealt with such situation in his paper. In this study, we break ties by randomly selecting the first species in the list of dominant species. The third reason is specifically for Meanwise Resolution. In the original k-RAP, McGovern used Euclidean norm mapping and multiplied α by $\sqrt{2}$ to create a new filtering for his algorithm. By doing that, he inadvertently filtered out most of the isolates, hence, rendering his Meanwise Resolution ineffective with α filtering. In our case, we decided to keep the same α value when performing Meanwise Resolution. Overall, we obtained good accuracy for Meanwise Resolution and the results showed that similar to Intersection, Meanwise is also responsive to α filtering.

From the results, we found that the level of precision increased as we applied α filtering, but at the same time, the number of returned isolates labeled as “Undefined” also increased. Table 5.4 shows the numbers of “Undefined” isolates obtained for each resolutions with different α values. These isolates are labeled “Undefined” because they do not have any neighbors whose Pearson correlation coefficients are greater than a given α threshold. Even though we agree with McGovern that Intersection Resolution is more responsive to α and it also yields the highest precision, using Intersection Resolution also means there will be isolates that do not get classified and returned as “Undefined.” After Intersection, Meanwise Resolution seems to be the second best candidate when using α filtering. With Meanwise, fewer isolates are labeled as “Undefined” while the overall accuracy still stays well above 70%.

Even though our modified k-RAP algorithm yielded slightly different results than the original k-RAP, the resolutions do not perform any worse. In fact, for all resolutions, the overall accuracy stays above 65% while the precision stays above 80%. Therefore, we think the modifications that we made to k-RAP are acceptable and we will continue to use them in our classifier.

5.4.2 Clustering & k-Nearest Neighbors

Comparing to McGovern’s original k-RAP algorithm, our modified version yields acceptable results when we evaluated them against the CPLOP library. This prompted us to evaluate the combination of pyro-DBSCAN with our modified k-RAP in the hope of improving our MST process. Figure 5.4 shows the overall accuracy when combining clustering using pyro-DBSCAN and k-RAP at $\text{minPts} = 3$ and $\alpha = 0.00$. In Figure 5.4a, all neighbors are weighted equally, while in Figure 5.4b, we applied a weighting scheme on the isolates, giving each neighbor a weight of $\frac{1}{d}$ where d is the distance from an isolate to the unknown isolate.

In both cases, the precision values for all resolutions stay above 73%. However, the weighted k-RAP tend to decrease the precision by 1-2%. The decrease in the overall accuracy is more pronounced, especially in Meanwise and Intersection. It suggests that Meanwise and Intersection are more susceptible to the weighted k-RAP algorithm.

Since combining clustering using pyro-DBSCAN and k-RAP gave us an overall accuracy above 70%, we wanted to go further and see whether we could attain an even better accuracy when we perform clustering and then use the resulting clusters to perform k-Nearest Clusters. This idea is evaluated in the next section.

5.4.3 Clustering & k-Nearest Clusters

The main goal of this thesis is to finalize a MST method that seeks to individually outperform both clustering by pyro-DBSCAN and k-RAP. Our final classifier combines both clustering using pyro-DBSCAN and k-RAP into one hybrid algorithm, which henceforth will be referred to as HAP - Hybrid Algorithms for Pyroprints. HAP is comprised of four different possible methods for assigning clusters’ properties:

(a) pyro-DBSCAN & k-RAP

(b) pyro-DBSCAN & weighted k-RAP

Figure 5.4: The overall accuracy when we combined pyro-DBSCAN and k-RAP at minPts=3, $\alpha = 0.00$.

- Unweighted dominant species selection method (DS)
- Unweighted collective species selection method (CS)
- Weighted dominant species selection method (wDS)
- Weighted collective species selection method (wCS)

Since each of the four HAP methods uses four k-RAP resolutions, together, we have 16 pairs of decision strategies total. For each pair, we evaluated the overall precision and overall accuracy with `minPts` = 3, 4 and 5 and $\alpha = \{0.0, 0.98, 0.99, 0.995\}$. Figures 5.5, 5.6, 5.7, 5.8 show the overall precision and the overall accuracy of our classifier when `minPts` = 3 and $\alpha = \{0.0, 0.98, 0.99, 0.995\}$. The overall accuracy is labeled as “Adjusted.” The similar results for `minPts` = 4 and 5 can be found in the Appendix B.

As we increase the `minPts` values, the overall accuracy slightly decreased across all pairs. This might have been due to the decrease in the number of clusters and the drop in clustering purity as the size of the clusters grow.

The overall precision and accuracy also changed as we varied the values of α . However, regardless of α , across all four k-RAP resolutions, DS shows the best results, CS shows the second best and then came wDS and wCS. The difference in the accuracy level between DS versus CS and wDS versus wCS can be explained by the dilution of the dominant species. We go from considering each cluster as a single dominant species in DS and wDS to considering each cluster as a collection of species in CS and wCS. Considering each cluster as a collection of species, in effect, lessens the significance of the dominant species in a cluster, which leads to a lower overall precision across all four resolutions (around 1-2%) in CS and wCS. When comparing DS versus wDS and CS versus wCS, the overall precision is also being slightly lowered (around 1%). This lowering is due to the use of the cluster weight coefficient. So far,

Table 5.5: The number of isolates that are returned as “Undefined” by our classifier for $\text{minPts} = 3$ and $\alpha = 0.00, 0.95, 0.98, 0.99$ and 0.995 .

α	0.00	0.95	0.98	0.99	0.995
Winner	0	0	2	9	81
Union	0	0	2	9	81
Meanwise	0	2	67	400	1076
Intersection	0	15	157	663	1306

the results suggest that it is better to assign a single dominant species to each cluster and to not use cluster weight coefficients.

As we increased α , the performance of each k-RAP resolution changed. Meanwise and Intersection Resolution started to perform better. This is likely due to the effect of α filtering, and selecting only isolates with high similarities of both ITS regions. However, the higher precision value also comes with a trade-off. There are also higher numbers of isolates that did not get classified, see Table 5.5.

At $\alpha = 0.00$, across all four HAP methods, the results (Figure 5.5) show that Union performed the best while Intersection performed the worst. When examining the drop of Union Resolution in DS versus CS and wDS versus wCS, we found that the majority of the misclassified isolates came from Cow. However, given that CPLOP is made up of almost 40% of Cow isolates, the chances of misclassifying Cow isolates is not surprising. The performance of Meanwise and Winner depended on each HAP method. In DS and CS methods, Meanwise performs slightly better than Winner while the reverse is true when the weight coefficient is applied in wDS and wCS. This suggests that Meanwise is slightly better than Winner when no weighting scheme is applied and vice versa. Calculating the overall drop in accuracy when going to DS to wDS and CS to wCS still shows that Meanwise and Intersection are more susceptible to the weighting scheme. However, since the difference is not very high, only around

Figure 5.5: The overall accuracy for four HAP methods with different resolutions at $\minPts=3$ and $\alpha = 0.00$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure 5.6: The overall precision and accuracy for four HAP methods with different resolutions at minPts=3 and $\alpha = 0.98$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure 5.7: The overall precision and accuracy for four HAP methods with different resolutions at minPts=3 and $\alpha = 0.99$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure 5.8: The overall precision and accuracy for four HAP methods with different resolutions at minPts=3 and $\alpha = 0.995$.

1-2%, the use of cluster weight coefficient does not have as high of an effect as we hoped that it would. This prompted us to investigate the behaviors of our classifier if we perform k-Nearest Clusters on only the true clusters, meaning clusters whose size > 1 . We named this *strict*-HAP to differentiate it with the regular HAP methods.

In *strict*-HAP, we only considered the neighboring clusters whose sizes were greater than one. Figure 5.9 shows the overall accuracy of *strict*-HAP. Even though the overall accuracy for all four *strict*-HAP methods is above 60%, compared to the regular HAP, it is lower by more than 10%. This drop in performance means that most of the isolates that did not get clustered by pyro-DBSCAN are really far away from any clusters. Hence, checking only the clusters around them while ignoring all other isolates is not a good solution. The drop in accuracy between CS and wCS compared to DS and wDS further demonstrates that diluting the clusters into several species significantly reduces the accuracy level.

5.4.4 Major & Under-Represented Species

Tables 5.6, 5.7, 5.8, and 5.9 show the precision, recall, and F_1 -score (P , R , and F) of some particular species obtained from HAP with four different resolutions at $k = 7$ and $\alpha = 0.00$.

Among all four HAP methods, at $\alpha = 0.00$, the overall values of P , R , and F for DS and wDS are closely similar and so are those of CS and wCS. Overall, DS and wDS shows a slightly higher accuracy than CS and wCS. In the DS method, Union seems to perform best, then came Meanwise and Winner. The Intersection Resolution shows the lowest P , R , and F values. However, it is worth noting that the differences among the four resolutions are not significant, only around a few percentiles.

For major species such as Cow and Human, the algorithm shows a high level of accuracy, while for under-represented species, their P , R , F values are a lot lower

Figure 5.9: The overall accuracy for four *strict*-HAP methods with different resolutions at $\text{minPts} = 3$, $\alpha = 0.00$.

Table 5.6: Precision (P), Recall (R), and F -Measure (F_1) of some particular species in CPLOP obtained from Unweighted Dominant Species selection method (DS) with four resolutions at $k = 7$, $\alpha = 0.00$.

Host Species	Isolates	Union			Winner		
		P (%)	R (%)	F_1	P (%)	R (%)	F_1
Overall	4785	77.6	77.6	0.776	76.93	76.93	0.769
Cow	1749	72.07	87.76	0.791	72.29	86.96	0.79
Human	1636	80.41	92.85	0.862	80.31	92.48	0.86
Pigeon	233	85.59	43.35	0.575	80.8	43.35	0.564
Wild Turkey	191	88.0	57.59	0.696	85.71	56.54	0.681
Dog	179	93.27	54.19	0.686	88.35	50.84	0.645
Seagull	78	96.88	79.49	0.873	95.38	79.49	0.867
Cat	46	85.0	36.96	0.515	89.47	36.96	0.523
Chicken	44	66.67	9.09	0.16	60.0	13.64	0.222
Bat	37	83.72	97.3	0.9	83.72	97.3	0.9
Bear	6	83.33	83.33	0.833	83.33	83.33	0.833
		Meanwise			Intersection		
Overall	4785	77.28	77.28	0.773	75.82	75.82	0.758
Cow	1749	74.61	84.33	0.792	72.44	83.7	0.777
Human	1636	79.85	93.03	0.859	78.97	92.05	0.85
Pigeon	233	73.83	47.21	0.576	72.41	45.06	0.556
Wild Turkey	191	82.27	60.73	0.699	79.41	56.54	0.661
Dog	179	85.25	58.1	0.691	84.48	54.75	0.664
Seagull	78	94.12	82.05	0.877	94.03	80.77	0.869
Cat	46	62.96	36.96	0.466	60.0	26.09	0.364
Chicken	44	40.0	13.64	0.203	26.32	11.36	0.159
Bat	37	83.72	97.3	0.9	83.72	97.3	0.9
Bear	6	83.33	83.33	0.833	83.33	83.33	0.833

Table 5.7: Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Unweighted Collective Species selection method (CS) with four resolutions at $k = 7$, $\alpha = 0.00$.

Host Species	Isolates	Union			Winner		
		P (%)	R (%)	F_1	P (%)	R (%)	F_1
Overall	4785	76.61	76.61	0.766	75.53	75.53	0.755
Cow	1749	73.63	84.62	0.787	73.54	82.79	0.779
Human	1636	79.13	92.73	0.854	78.88	91.99	0.849
Pigeon	233	76.55	47.64	0.587	71.34	48.07	0.574
Wild Turkey	191	85.71	56.54	0.681	82.44	56.54	0.671
Dog	179	83.62	54.19	0.658	81.42	51.4	0.63
Seagull	78	96.88	79.49	0.873	96.88	79.49	0.873
Cat	46	72.73	34.78	0.471	76.19	34.78	0.478
Chicken	44	80.0	9.09	0.163	66.67	13.64	0.226
Bat	37	73.47	97.3	0.837	73.47	97.3	0.837
Bear	6	75.0	100.0	0.857	54.55	100.0	0.706
		Meanwise			Intersection		
Overall	4785	75.84	75.84	0.758	74.34	74.34	0.743
Cow	1749	76.1	81.19	0.786	73.71	79.99	0.767
Human	1636	78.92	92.91	0.853	77.77	91.93	0.843
Pigeon	233	62.21	45.92	0.528	60.95	44.21	0.512
Wild Turkey	191	74.68	60.21	0.667	76.81	55.5	0.644
Dog	179	79.53	56.42	0.66	78.05	53.63	0.636
Seagull	78	91.55	83.33	0.872	90.0	80.77	0.851
Cat	46	68.0	36.96	0.479	70.59	26.09	0.381
Chicken	44	42.86	13.64	0.207	31.25	11.36	0.167
Bat	37	72.0	97.3	0.828	75.0	97.3	0.847
Bear	6	35.71	83.33	0.5	29.41	83.33	0.435

Table 5.8: Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Weighted Dominant Species selection method (wDS) with four resolutions at $k = 7$, $\alpha = 0.00$.

Host Species	Isolates	Union			Winner		
		P (%)	R (%)	F_1	P (%)	R (%)	F_1
Overall	4785	77.55	77.55	0.776	76.36	76.36	0.764
Cow	1749	72.04	87.65	0.791	71.83	86.45	0.785
Human	1636	80.24	92.85	0.861	79.51	92.48	0.855
Pigeon	233	87.93	43.78	0.585	80.49	42.49	0.556
Wild Turkey	191	87.1	56.54	0.686	85.12	53.93	0.66
Dog	179	95.05	53.63	0.686	85.05	50.84	0.636
Seagull	78	96.88	79.49	0.873	96.88	79.49	0.873
Cat	46	81.82	39.13	0.529	77.27	36.96	0.5
Chicken	44	66.67	9.09	0.16	66.67	9.09	0.16
Bat	37	83.72	97.3	0.9	83.72	97.3	0.9
Bear	6	83.33	83.33	0.833	83.33	83.33	0.833
		Meanwise			Intersection		
Overall	4785	75.82	75.82	0.758	74.71	74.71	0.747
Cow	1749	73.53	83.08	0.78	71.5	83.19	0.769
Human	1636	79.23	92.54	0.854	78.25	91.69	0.844
Pigeon	233	69.08	45.06	0.545	69.86	43.78	0.538
Wild Turkey	191	75.84	59.16	0.665	74.63	52.36	0.615
Dog	179	76.0	53.07	0.625	78.63	51.4	0.622
Seagull	78	89.86	79.49	0.844	92.54	79.49	0.855
Cat	46	61.54	34.78	0.444	63.16	26.09	0.369
Chicken	44	26.67	9.09	0.136	27.78	11.36	0.161
Bat	37	83.72	97.3	0.9	83.72	97.3	0.9
Bear	6	83.33	83.33	0.833	83.33	83.33	0.833

Table 5.9: Precision (P), Recall (R), and F -Measure (F_1) of some major species in CPLOP obtained from Weighted Collective Species selection method (wCS) with four resolutions at $k = 7$, $\alpha = 0.00$.

Host Species	Isolates	Union			Winner		
		P (%)	R (%)	F_1	P (%)	R (%)	F_1
Overall	4785	76.78	76.78	0.768	75.57	75.57	0.756
Cow	1749	72.77	85.25	0.785	72.8	84.16	0.781
Human	1636	79.04	92.91	0.854	78.43	92.24	0.848
Pigeon	233	79.7	45.49	0.579	72.34	43.78	0.545
Wild Turkey	191	87.6	55.5	0.679	84.03	52.36	0.645
Dog	179	83.19	55.31	0.664	79.66	52.51	0.633
Seagull	78	96.88	79.49	0.873	96.88	79.49	0.873
Cat	46	84.21	34.78	0.492	76.19	34.78	0.478
Chicken	44	80.0	9.09	0.163	80.0	9.09	0.163
Bat	37	73.47	97.3	0.837	73.47	97.3	0.837
Bear	6	83.33	83.33	0.833	50.0	83.33	0.625
		Meanwise			Intersection		
Overall	4785	75.53	75.53	0.755	74.11	74.11	0.741
Cow	1749	74.82	81.53	0.78	72.82	80.73	0.766
Human	1636	78.64	92.73	0.851	77.29	92.18	0.841
Pigeon	233	65.82	44.64	0.532	65.79	42.92	0.519
Wild Turkey	191	75.16	60.21	0.669	75.19	52.36	0.617
Dog	179	74.81	54.75	0.632	75.0	51.96	0.614
Seagull	78	88.73	80.77	0.846	88.73	80.77	0.846
Cat	46	66.67	34.78	0.457	66.67	26.09	0.375
Chicken	44	40.0	9.09	0.148	33.33	11.36	0.169
Bat	37	72.0	97.3	0.828	75.0	97.3	0.847
Bear	6	41.67	83.33	0.556	38.46	83.33	0.526

than the overall values. This is certainly due to their lower number of instances in the database. It again shows that in a library-dependent MST, the accuracy level of a classification process is strongly dependent on a species' representation in the library.

Among all the under-represented species, Bat and Bear seem to show some exceptions. We expected the P , R , and F for Bat to be similar or lower than that of Chicken since Bat has a lower number of instances than Chicken. However, Bat's P , R , and F values turned out to be higher than Chicken and even higher than Seagull, who has more than double Bat's number of instances in the database. One explanation for this can be due to Bat's uniqueness in its *E. coli* strains which helps segregate Bat from other host species. The other exception is Bear. Even though it has only six isolates in the database, we obtained high values for its P , R , and F across all four HAP methods and all four resolutions. This either means that the *E. coli* strains from Bear must also be unique or Bear is simply an outlier. The latter one is favored since we also performed the same test on other isolates with a similar number of instances in the database and we ended up with zero values for all evaluation metrics.

Chapter 6

CONCLUSION

The Cal Poly Library of Pyroprints or CPLOP is a tool built to help identify the source of fecal contamination in the water supplies in the San Luis Obispo area. It uses *E. coli* as the fecal indicator bacteria. The fingerprints used in the classification process are obtained from pyroprinting, a method developed by the researchers at the Biology Department at Cal Poly. These fingerprints of known samples are stored and managed in CPLOP.

The work in this thesis is developed to be used with CPLOP. The main contribution of this thesis is a hybrid classifier which is termed HAP - Hybrid Algorithm for Pyroprints. HAP provides a set of 16 decision strategies resulting from the combination of the clustering algorithm (pyro-DBSCAN) and k-RAP.

In addition to HAP, this thesis has the following contributions:

- An evaluation of clustering using pyro-DBSCAN in terms of entropy,
- An empirical study measuring the classification accuracy of HAP using the samples stored in CPLOP, and
- An evaluation of the classification accuracy of HAP, compared to the individual performance of pyro-DBSCAN and k-RAP.

6.1 Clustering Entropy

In our study, we consider each cluster resulting from clustering using pyro-DBSCAN as a bacterial strain. Clustering entropy indicates the degree of disorder in a cluster. The closer the value of entropy to 0, the better the clustering algorithm. In other

words, the closer the value of entropy to 0, the purer are the bacterial strains created by pyro-DBSCAN.

As we varied `minPts` from 1 to 9, we obtained an average entropy of 0.55. This value indicates that our clustering algorithm did a good job at discerning amongst bacterial strains.

Evaluation done on clustering purity by McGovern’s also shows promising results for pyro-DBSCAN [8]. Since pyro-DBSCAN is quite effective at discriminating amongst strains, we make the decision of using clusters resulting from pyro-DBSCAN in the first step of our 2-step classification technique. However, since pyro-DBSCAN only clusters about 40% of isolates, in order to make sure every isolate will get classified, for the second step, we apply the k-Nearest Clusters Algorithm.

6.2 Accuracy and Effectiveness of HAP

HAP provides a total set of 16 different decision strategies for host species classification. These strategies come from four different HAP methods: DS, CS, wDS, wCS and four k-RAP resolutions: meanwise, winner, union, intersection.

When evaluating at `minPts` = 3 and no filtering, Union showed the best precision across all four HAP methods, the average precision values are 77.9%, 76.7%, 77.7% and 76.8% for DS, CS, wDS, and wCS respectively. Meanwise and Winner showed very similar precision values with the exception of the wDS method where Winner’s precision is higher than Meanwise by 1%. Intersection showed the worst performance with the highest precision value of 76% for DS method and around 74.7%, 74.7% and 74% for CS, wDS and wCS.

As we increased α filtering, the performance of each decision strategy changed. If we consider only the precision, then Intersection showed the best results across all

four HAP methods: above 75% when $\alpha = 0.98$ and above 80% when $\alpha = 0.99$ and 0.995. However, when taking into account all the isolates that did not get classified, the overall accuracy of Intersection dropped to 74%, 71% and 63% when $\alpha = 0.98$, 0.99 and 0.995, respectively. This drop is due to the high number of isolates that failed to get classified due to α -filtering. Meanwise’s performance also drops with α -filtering. Before adjustment, the average precision values were 77.3%, 81%, and 86.2%. After adjusting for all unclassified isolates, these numbers dropped to 76.3%, 74.2% and 66.9% for when $\alpha = 0.98$, 0.99 and 0.995, respectively.

While the adjusted accuracy for Intersection and Meanwise dropped as we increased α , the performance of Union and Winner did not change as much. They still stayed relatively high and above 75%.

From our evaluation, we can conclude that among all four HAP methods, the order of performance is DS, wDS, CS and then wCS. In each individual HAP method, the performance of the k-RAP resolutions is highly dependent on α -filtering and whether or not we take into account the number of unclassified isolates. This is a trade-off that one must make when deciding on which pair of classifier to use. If no filtering is required, then DS-Union is the best candidate. In the case of α filtering, if one prefers a high precision level, then DS-Intersection is a good candidate. In the case where DS-Intersection fails to give a final classification, one can opt for DS-Union or DS-Meanwise, with or without filtering, to obtain a final classification.

6.3 Future Work

6.3.1 Incorporate HAP into CPLOP

The current CPLOP is using OHClust! as its clustering algorithm for strain differentiation. As shown in previous work, OHClust! required far more resources than

the CPLOP server can handle. In addition, the strains produced by OHClust! did not exactly match the biologist’s notion of strains. Since HAP performance showed significant improvement over other methods, future work needs to be done to incorporate HAP into CPLOP directly. We should also provide an interactive way for researchers to select which decision strategy they prefer.

Since the effectiveness of our classifier is highly dependent on the database and its size, in order to increase the accuracy level, especially for under-represented species, it is important to expand the library with more species.

6.3.2 Amazon Web Services - AWS

The current website is hosted by the Cal Poly servers, which presents many challenges. The first challenge is the limitation in computational ability. The CPLOP server contains only 4GB of RAM. Secondly, there is currently no designated point of contact for server administration. Moving CPLOP server to a more dynamic and scalable platform like AWS would address not only the computational constraint of the current server’s hardware but also allow for multiple parties to do server administration and/or site enhancements.

AWS is a web service provided by Amazon on a paid subscription basis. With AWS, we can add computing resources as needed. The monthly bill is also based on the amount of computing power and resources required. Migrating the CPLOP website to AWS also makes it easier to update and administer as well as make use of aspects of AWS infrastructure that will make for a secure, faster experience for researchers. The CPLOP website can be completely migrated to AWS, but for a fully revamped experience, the application layer of the CPLOP website must also be rewritten using more modern frameworks and libraries to address potential application security issues and performance bottlenecks.

BIBLIOGRAPHY

- [1] Microbial Source Tracking - FAQs.
<http://www.biovir.com/Images/pdf072.pdf>. Accessed: 2018-01-10.
- [2] Real Statistics Using Excel. <http://www.real-statistics.com/correlation/basic-concepts-correlation/>. Accessed: 2018-01-18.
- [3] Stattrek. <http://stattrek.com/statistics/dictionary.aspx?definition=z%20score>. Accessed: 2018-01-18.
- [4] 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.
- [5] A. Dufour and J. Bartram. *Animal waste, water quality and human health*. IWA Publishing, 2012.
- [6] A. Hillebrand, R. Wurm, A. Menzel, and R. Wagner. The seven e. coli ribosomal rna operon upstream regulatory regions differ in structure and transcription factor binding efficiencies. *Biological chemistry*, 386(6):523–534, 2005.
- [7] 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.

- [8] J. D. McGovern. Investigating the k-Nearest Neighbors Resolution Algorithms for Pyroprints and Clustering for Bacterial Strains. Master's thesis, California Polytechnic State University, San Luis Obispo, 2016.
- [9] A. Montana. Algorithms for Library-based Microbial Source Tracking. Master's thesis, California Polytechnic State University, San Luis Obispo, 2013.
- [10] W. E. C. on Malaria and W. H. Organization. *WHO expert committee on malaria: twentieth report*. Number 892. World Health Organization, 2000.
- [11] F. A. Rainey, N. L. Ward-Rainey, P. H. Janssen, H. Hippe, and E. Stackebrandt. Clostridium paradoxum dsm 7308t contains multiple 16s rrna genes with heterogeneous intervening sequences. *Microbiology*, 142(8):2087–2095, 1996.
- [12] 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.
- [13] D. Shealy. Exploration of PyroPrinting for Environmental Forensics. Technical report, California Polytechnic State University, San Luis Obispo, California, June 2012.
- [14] J. L. Soliman. CPLOP: The Cal Poly Library of Pyroprints. Master's thesis, California Polytechnic State University, San Luis Obispo, 2013.

APPENDICES

Appendix A

PURITY

Purity measures the extent that a cluster contains only one class of data. For each cluster, the purity is computed as:

$$purity(D_i) = \max_j (Pr_i(c_j)) \quad (\text{A.1})$$

where $Pr_i(c_j)$ is the proportion of class c_j data points in cluster D_i .

The total purity considering all clusters is:

$$purity_{total}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} * purity(D_i) \quad (\text{A.2})$$

where $|D_i|$ is the total number of data points in cluster D_i , $|D|$ is the total data points, and $purity(D_i)$ is the purity of cluster D_i .

For example:

Cluster	Cow	Dog	Human	Total	Entropy
1	31	0	1	32	0.2
2	0	0	5	5	0.0
3	10	1	2	13	0.991
Total	41	1	8	50	0.385

Figure A.1: A simple example showing how to calculate the clustering purity. $purity_1 = \max(\frac{31}{32}, \frac{1}{32}) = \frac{31}{32} = 0.969$

$$purity_2 = \max(\frac{5}{5}) = 1$$

$$purity_3 = \max(\frac{10}{13}, \frac{1}{13}, \frac{2}{13}) = \frac{10}{13} = 0.769$$

$$Entropy_{total} = 0.969 * \frac{32}{50} + 1 * \frac{5}{50} + 0.769 * \frac{13}{50} = 0.92$$

Appendix B

HAP - HYBRID AGORITHM FOR PYROPRINTS

B.1 `minPts = 4`

B.2 `minPts = 5`

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.1: The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.00$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.2: The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.98$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.3: The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.99$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.4: The overall precision for four HAP methods with different resolutions at minPts = 4 and $\alpha = 0.995$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.5: The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.00$.

Figure B.6: The overall precision for four HAP methods with different resolutions at $\minPts = 5$ and $\alpha = 0.98$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.7: The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.99$.

(a) DS

(b) CS

(c) wDS

(d) wCS

Figure B.8: The overall precision for four HAP methods with different resolutions at minPts = 5 and $\alpha = 0.995$.