In search of more stable hierarchical trees: devising new algorithms to improve upon the stability of *neighbor joining*

by

Amanda Michelle Saunders

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements for the degree of
Master of Science in
Bioinformatics

Guelph, Ontario, Canada

©Amanda Michelle Saunders, June, 2017
IN SEARCH OF MORE STABLE HIERARCHICAL TREES: DEVISING NEW
ALGORITHMS TO IMPROVE UPON THE STABILITY OF NEIGHBOR JOINING

Amanda Saunders
University of Guelph, 2017

The standard method of constructing hierarchical trees – neighbour joining – while commonly used, has some major flaws. For most data sets not derived from a common descent process, it produces trees that are highly unstable. Adding or deleting a point can cause dramatic shifts in the tree topology. The degree of this shift can be measured by calculating the distance between trees. To provide an alternative to neighbour joining, bubble clustering was devised. Instead of using the simple Euclidean distance between points, the connections between points are established by repeated sampling. A multi-dimensional sphere is placed into the data space and the association between all points within the sphere is increased. Once this is complete the resultant matrix is converted into a tree by sequentially joining the most associated points. The new algorithm proved to generate more stable trees compared to neighbour joining when applied to random or real data.
ACKNOWLEDGEMENTS

I would like to thank my wonderful fiance and his family for always supporting me and helping me even though my work is very much outside of their areas of expertise.

I would also like to thank my supervisor, Daniel Ashlock, for introducing me to a highly intriguing problem.
# Table of Contents

List of Tables vii

List of Figures viii

1 Bioinformatics and Clustering 1

1.1 Introduction ................................................. 1
1.2 Hierarchical Clustering ................................. 2
1.3 Role of Hierarchical Clustering in Bioinformatics ....... 3
1.4 Stability and Why It Matters ............................. 4
1.5 Goals ......................................................... 7

2 Clustering Today 10

2.1 Clustering Terminology ................................. 10
2.2 Clustering Algorithms ..................................... 11
2.2.1 Connectivity-based clustering .......................... 12
   Mechanics ................................................. 12
   Example Algorithms ...................................... 12
   Benefits .................................................. 12
   Drawbacks ............................................... 13
2.2.2 Centroid-based clustering ............................ 13
   Mechanics ................................................. 13
   Example Algorithms ...................................... 14
   Benefits .................................................. 14
   Drawbacks ............................................... 15
2.2.3 Model based clustering ................................ 16
   Mechanics ................................................. 16
   Example Algorithms ...................................... 16
   Benefits .................................................. 16
   Drawbacks ............................................... 17
2.2.4 Density-based clustering ............................. 17
6 Future directions 85
6.1 Exploration of algorithm details 86
6.2 Improvements to the stability measure 87
6.3 Applying bubble clustering to different data sets 87
6.4 Examination of behaviour on problematic data sets 88
6.5 Further Applications of Stability Measure and Tree Distance Metric 88

References 90
List of Tables

2.1 Constructed data set used to illustrate Ashlock et al.’s (Ashlock et al. (2009)) method for testing stability ......................................................... 25

2.2 MCCs for each pair of points in the Snip and Rebuild trees with the 5th sample removed ................................................................. 33

3.1 Example associator matrix ................................................................. 38

4.1 Variance of Stability Scores of bubble clustering trees in Figures 4.11 and 4.12 ................................................................. 55
List of Figures

1.1 An example of a hierarchical clustering in the form of a tree .......................... 3
1.2 Generating the dendrogram for the snip tree with the first sample removed. .. 6

2.1 Tree using hclust() using the data in Table 2.1 ................................. 26
2.2 Both trees represent dendrograms of the data in Table 2.1 with the 1st sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 27
2.3 Both trees represent dendrograms of the data in Table 2.1 with the 2nd sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 28
2.4 Both trees represent dendrograms of the data in Table 2.1 with the 3rd sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 29
2.5 Both trees represent dendrograms of the data in Table 2.1 with the 4th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 30
2.6 Both trees represent dendrograms of the data in Table 2.1 with the 5th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 31
2.7 Both trees represent dendrograms of the data in Table 2.1 with the 6th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom. ............................................................... 32

3.1 Tree built from the example associator matrix in table 3.1 ......................... 38
3.2 Examples of a Snip and a Rebuild tree generated using hclust(method=complete) on random data ......................................................... 40

4.1 Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 2 observations randomly generated between -5 and 5. Label description in the text. ......................... 48
4.2 Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 5 observations randomly generated between -5 and 5. Label description in the text.

4.3 Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 10 observations randomly generated between -5 and 5. Label description in the text.

4.4 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 10x50 bubbles (B1).

4.5 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 100x50 bubbles (B2).

4.6 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 1000x50 bubbles (B3).

4.7 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the complete linkage method (H1).

4.8 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the average method (H2).

4.9 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the centroid method (H3).

4.10 Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with n=25, 50 and 100 data points observations randomly generated between -5 and 5. To allow comparison between different sized data sets resultant stability per Snip-Rebuild pair was normalized so that the results of the 25, 50 and 100 runs all individually fit in the range of [0,1].

4.11 Average stability per Snip-Rebuild pair results on the Problem Taxonomy data set n=34 with 10 runs each.

4.12 Average stability per Snip-Rebuild pair results on the Problem Taxonomy data set n=34 with 15 runs each.

4.13 Two example trees of the problem taxonomy data set constructed using bubble clustering with $10^4 n$ bubbles.
4.14 Hierarchical tree of the problem taxonomy data set constructed using bubble clustering with $10^5 n$ bubbles. .............................. 58
4.15 Hierarchical tree of the problem taxonomy data set constructed using hclust(method=complete). .............................. 58
4.16 Snip hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with bubble clustering with $10^4 n$ bubbles. .............................. 59
4.17 Rebuilt hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with bubble clustering with $10^4 n$ bubbles. .............................. 60
4.18 Snip hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with hclust(method=complete). .............................. 60
4.19 Rebuilt hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with hclust(method=complete). .............................. 60
4.20 Average stability of Snip-Rebuild pairs generated from re-ordered versions of the problem taxonomy data set (see text for description) .............................. 62
4.21 Comparison of average stability of the real problem taxonomy data set and re-ordered versions .............................. 63
4.22 Average stability of Snip-Rebuild pairs generated from first of the DNA dimer data sets n= 50 with 32 runs .............................. 64
4.23 Average stability of Snip-Rebuild pairs generated from second of the DNA dimer data sets n= 50 with 15 runs .............................. 65
4.24 Average stability of Snip-Rebuild pairs generated from third of the DNA dimer data sets n= 50 with 15 runs .............................. 65
4.25 Hierarchical tree of the dimersA data set constructed using bubble clustering with $10^5 n$ bubbles. .............................. 66
4.26 Hierarchical tree of the dimersA data set constructed using hclust(method=complete). .............................. 67
4.27 Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^5 n$ bubbles. .............................. 67
4.28 Hierarchical tree of the dimersB data set constructed using hclust(method=complete). .............................. 68
4.29 Hierarchical tree of the dimersC data set constructed using bubble clustering with $10^5 n$ bubbles. .............................. 68
4.30 Hierarchical tree of the dimersC data set constructed using hclust(method=complete). .............................. 69
4.31 Hierarchical tree of the dimersA data set constructed using bubble clustering with $10^5 n$ bubbles with sample 48 snipped. .............................. 70
4.32 Hierarchical tree of the dimersA data set constructed using bubble clustering with $10^5 n$ bubbles with sample 48 removed from the data set. .............................. 70
4.33 Hierarchical tree of the dimersA data set constructed using hclust(method=complete) with sample 48 snipped. .............................. 71
4.34 Hierarchical tree of the dimersA data set constructed using hclust(method=complete) with sample 48 removed from the data set. .............................. 72
4.35 Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^5 n$ bubbles with sample 10 snipped. .............................. 72
4.36 Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^5 n$ bubbles with sample 10 removed from the data set. .............................. 73
4.37 Hierarchical tree of the dimersB data set constructed using hclust(method=complete) with sample 10 snipped. ......................................................... 73
4.38 Hierarchical tree of the dimersB data set constructed using hclust(method=complete) with sample 10 removed from the data set. .............................................. 74
4.39 Hierarchical tree of the dimersC data set constructed using bubble clustering with $10^5n$ bubbles with sample 3 snipped. .................................................. 74
4.40 Hierarchical tree of the dimersC data set constructed using bubble clustering with $10^5n$ bubbles with sample 3 removed from the data set. ............... 75
4.41 Hierarchical tree of the dimersA data set constructed using hclust(method=complete) with sample 3 snipped. ................................................................. 75
4.42 Hierarchical tree of the dimersC data set constructed using hclust(method=complete) with sample 3 removed from the data set. ................................. 76
Chapter 1

Bioinformatics and Clustering

1.1 Introduction

Bioinformatics is the discipline that has emerged to deal with the problems generated by advances in data collection that allow for the amassing of enormous amounts of biological data. Bioinformatics is the science of processing this large amount of data into understandable bits of information. A readout of the gene activity profile of 100 samples may not have any easily noticeable patterns. Clustering simplifies this process. Being able to sort the 100 samples into well-defined categories and then comparing these groupings to previously collected data allows researchers to draw conclusions. If those 100 samples can be classified into three distinct groups, and if these clusters then can be related back to treatment data, we can confirm or refute hypotheses about the effects of those treatments more easily. Alternatively we can also cluster the gene activity readouts to try to identify
genes that operate together in response to a specific stimulus.

Unfortunately, many of the existing clustering algorithms are ill-equipped to deal with the high dimensional data that we have the technology to generate. This is in part due to the *curse of dimensionality* - as more and more dimensions are present in a data set the Euclidean distance metric between points increasingly loses significant information as more and more points become nearly equidistant from each other (Parsons et al. (2004)). This makes new research to generate improvements imperative to move the field forward. In addition, not all of the attributes of the data are relevant for cluster identification with one or more of the dimensions of the data only generating noise and impairing proper clustering (Agrawal et al. (1998)). Some of the available algorithms attempt to deal with this problem.

### 1.2 Hierarchical Clustering

Hierarchical clustering is a way of repeatedly grouping samples into clusters of similar data until all of the it is linked together with the end result represented by a hierarchical tree e.g. Figure 1.1. A hierarchical tree is a representation of the relationships between data items in a data set. Each data item is represented in a leaf of the tree and the leaves or taxa are connected together such that data items that are part of the same sub-tree have more in common with each other than with the data items not in the sub-tree. It is hierarchical in nature because the tree can be broken down into various different levels of sub-tree each representing a cluster within the data.
1.3 Role of Hierarchical Clustering in Bioinformatics

Clustering is, at its core, an attempt to separate a large number of data objects into a finite and understandable number of groupings that are based on some, possibly hidden, structure present in the data set. It becomes more and more important as the amount of data increases and generalizations become necessary in order to understand the nature of the data set (Xu and Wunsch (2010)). Given the immense amount of information available to today’s researchers, either through the new generation of data or examining existing records, some degree of data processing is required to be able to extract useful conclusions from these immense data sets. Since the completion of the Human Genome Project, there has been a large degree of interest in pulling useful information from large, often publicly available, data sets (Liew et al. (2005)). Early on, the limit on sequencing research was
the lack of fast computers, algorithms and public repositories of discovered sequences that were necessary in order for the field to reach its potential (Liew et al. (2005)). In some disciplines, like biomedical research, clustering has become ubiquitous, being used to analyze everything from gene expression data, to document mining to analyzing MRI data (Xu and Wunsch (2010)). As evidence of this, the number of published papers in Life and Health Sciences that rely on clustering analysis has continued to increase throughout the 2000s (Xu and Wunsch (2010)).

There are now entire areas of study that cannot be pursued without some form of clustering technology. For example, the entire field of metagenomics involves sequencing environmental DNA e.g. microbial DNA extracted from a soil sample (Roumpeka et al. (2017)). Without clustering this data it is a meaningless pile of genes. What can be done with clustering is get an idea of what is present in the original sample. Sequences are clustered into operational-taxonomic-units (OTUs) and from there, studies of diversity, gene expression, community structure and evolutionary relationships may be conducted (Roumpeka et al. (2017)).

1.4 Stability and Why It Matters

Often when clustering techniques are applied to data the true relationships are poorly understood and the quality of results cannot be measured by comparing the answer the applied algorithm generates to known relationships in the data (Parsons et al. (2004)). If
the true relationships were known then there would be no need to cluster the data. In the context of hierarchical clustering, ideally, if a tree was generated using only a partial data set the internal structure of that tree - how the taxa within it are related - would not be altered from the structure if the entire data set were to be used.

To measure the stability of trees we need a way to calculate the differences between different trees. There are a couple of ways in which this can be done. Fowlkes and Mallow’s method (Fowlkes and Mallows (1983)), for example, involves sequentially cutting the trees into more and more clusters while measuring the similarity/differences in cluster composition. Another method that has been previously mentioned is Ashlock et al.’s (Ashlock et al. (2009)) method of comparing vectors containing the minimum sub-tree sizes for pairs of points.

To test and compare hierarchical trees generated using neighbour joining and bubble clustering, tree stability is to be calculated using Ashlock et al.’s (Ashlock et al. (2009)) method. Trees are rebuilt using the original data with a sample removed and compared to the tree built from the full data set with the corresponding leaf directly removed from the tree (See Figure 1.2 for an example). A stability measure is generated by finding the smallest sub-tree that contains every single pair of data items and then comparing the distance between the two lists of sub-tree sizes for all possible sample deletions. The more similar the two trees, the smaller the distance between the two lists of smallest sub-trees. If the two trees are widely different then the tree constructed with all the data points is said to be unstable.
Figure 1.2: Generating the dendrogram for the snip tree with the first sample removed.
1.5 Goals

The research goals of this thesis are as follows: to try to improve upon previous methods of hierarchical clustering we have devised a different way of measuring the level of association between points. Instead of simply calculating a Euclidean distance between points we are instead measuring their level of association. The associator matrix is built through repeated measures of association of pairs of points on the data set and awarding points that pass the association check a reward and updating the matrix. The main benefit of this is that it is a better representation of how points are related as the effect of any one point on the ending topology of the tree is greatly reduced.

Bubble clustering is one such way of measuring the level of association between points. Bubble clustering’s measure of association is whether points are within the same randomly placed sphere. The distribution from which a radius for the sphere is selected is determined by the size of the data space. The initial pseudorandom number is generated using 48-bit arithmetic and returned as a number between 0.0 and 1.0. This is then multiplied by the difference between the largest and smallest distances between points in the data set followed by the addition of the value of the smallest distance between data points. Centres of the random spheres are selected from the data points and all points that fall within the sphere are given an association reward. This is performed multiple times to build up an associator matrix that is an accurate representation of the relationships within the data. Coded into the algorithm is the ability to select the desired number of bubbles used to build the associator matrix proportional to the number of points in the data set. Because of the
way it is constructed, the associator matrix generated using bubble clustering contains information on how often points are judged to be in the same general vicinity which, while related to the Euclidean distance, is not identical. The association measure between points obtained at the end is dependent on the density of points in their neighbourhood, but Euclidean distance between two points in multi-dimensional space is the same regardless of the content of the surroundings.

The goal of this new algorithm, as shown in Algorithm 1, is to generate more stable hierarchical trees than previous methods and release the finished algorithm as a software package for R.

**Algorithm 1 Bubble clustering - construction of associator matrix**

**Input:** Numeric data matrix with n samples

degree of bubble coverage(L)

**Output:** Associator matrix

**Details:**

Initialize n x n associator matrix with 0.000000001s

Repeat

Select random center (c) from n samples

Generate random radius (r)

Number in bubble (N) set to 0

For i in n
IF $\text{dist}(i, c) < r$ Increment N by 1

For $i$ in $n$

For $j$ in $n$

IF $\text{dist}(i, c) < r$ AND $\text{dist}(j, c) < r$ associator matrix $(i,j) += 1/N$

Until $10^L n$ iterations have been performed

Return: associator matrix
Chapter 2

Clustering Today

2.1 Clustering Terminology

The myriad of different ways to cluster data can be broadly divided into several general categories. Two major categories are *direct or partitional clustering* and *hierarchical clustering* (Agrawal et al. (1998)). Direct clustering divides the data into a static number of clusters with each cluster containing some of the data sets, whereas hierarchical clustering, as the name suggests, involves a nested division of the data into 1, 2, ... n clusters where n is the number of data points in the data set (Agrawal et al. (1998)).

One of the other major divisions possible when discussing clustering algorithms is the difference between bottom-up algorithms and top-down algorithms. Bottom-up algorithms start with all the data points in their own clusters and sequentially join them until the desired number is reached whereas top-down methods start with the data in a single cluster and it
is sequentially divided into more clusters until the desired number is reached.

2.2 Clustering Algorithms

There are many different ways to approach clustering, mostly because are multiple different ways to define what delineates the boundaries between clusters. Existing algorithms can also be tinkered with to improve performance, either in the stability of the results that they provide or in an attempt to shorten the run-time without sacrificing the power of the original algorithm. The combination of these two factors leads to the existence of a multitude of different algorithms that all at their core are for processing a large amount of data into easily digestible results that can be used to draw conclusions. A brief synopsis of various methodologies and algorithms follows.

There are also many different ways to determine closeness between clusters that contain multiple points: distance from the cluster means, distance from the cluster centroids, largest distance between points in cluster A and cluster B, the shortest distance between points in cluster A and cluster B etc. with each of theses different approaches generating different algorithms.
2.2.1 Connectivity-based clustering

Mechanics

Clustering is generated by using information on how points are connected e.g. distance between data points in order to group together those that are closest together.

Example Algorithms

The most commonly used algorithm for hierarchical clustering is variants of the *neighbour joining* algorithm. Neighbour joining refers to a class of agglomerative clustering algorithms that start with all data points in single clusters, identify the closest pair of data objects and join them and repeat until there is a single cluster. The difference between algorithms resulting from different methods for calculating the value of the newly formed sub-tree for use in subsequent searches for the closest pair (Ashlock et al. (2009); Murtagh and Contreras (2012)). The algorithm was developed to be able to generate minimum-evolution trees (Saitou and Nei (1987)) i.e minimizing the total branch length.

Benefits

Neighbour joining’s main strength is that it generates even large phylogenetic trees much quicker than other older methods (Tamura et al. (2004)).

The main benefit of connectivity based clustering, in general, is that it provides highly detailed information about relationships within the data. Unlike with other methods of clustering, which generally only provide whether or not two points are in the same cluster...
hierarchical clustering contains information on how relatively close two points are compared to the rest of the data set or even compared to the rest in the same neighbourhood.

**Drawbacks**

The main drawback of this method is that it is very sensitive to the loss or addition of data points: the produced tree can change dramatically if even a single point is removed from the data set. While the accuracy of neighbour joining has been tested for use in building hierarchical trees using phylogenetic data (Tamura et al. (2004); Desper and Gascuel (2004); Gascuel and Steel (2006)), the methods employed to try to accomplish this are very specific to the type of data being tested and are not focused on the performance of the algorithms in general. This is perhaps an oversight as the algorithm is not necessarily restricted to phylogenetic data sets. Most data sets not derived from common descent processes produce highly unstable trees when neighbour joining is applied (Ashlock et al. (2009)).

**2.2.2 Centroid-based clustering**

**Mechanics**

Centroid-based clustering breaks data sets into $k$ clusters by selecting $k$ points as centres so that the distance between the centres and all points in a cluster are minimized (Kanungo et al. (2002); Kleinberg (2002)). Different but related algorithms select different measurements to minimize.
Example Algorithms

- **k-means Clustering**
  - $k$ points are selected as centres and all other points assigned to cluster with closest centre followed by shifting the cluster centres. Points are reassigned to clusters and centers are shifted until there are no changes in center location (Kanungo et al. (2002))
  - The measurement being minimized is the mean squared error between the points in a $k$-cluster and its respective centre. (Kanungo et al. (2002); Kleinberg (2002))

- **k-median Clustering**
  - minimizes the sum Euclidean distance between the $k$ centres and the data points in their clusters (Kleinberg (2002))

- **k-centres Clustering**
  - minimizes the maximum distance between data points and the center of its cluster. (Kleinberg (2002))

**Benefits**

K-means and its derivatives are relatively simple algorithms and this allows them to be applied to a wide variety of problems without issue and also due to its simplicity it is a
relatively fast algorithm (Zhang et al. (2008)). In addition, the algorithm given sufficient run-time is able to find a local optimum in most instances (Zhang et al. (2008)).

**Drawbacks**

As with any algorithm that requires user input for the number of clusters present in the data, this requires a substantial amount of background knowledge to be confident that the number of clusters selected is the appropriate one. In addition, the basic versions of k-means are not very good at generating clusters with complex multidimensional shapes (Zhang et al. (2008); Kim et al. (2009)). k-means and other centroid-based clustering have a bias towards generating spherical clusters as they have a centre to which the remaining data points are assigned (Handl et al. (2005)). Another drawback of centroid-based clustering algorithms is that the initial seed of cluster centres can affect the final results, although this can be combated by selecting optimum starting centres (Zhang et al. (2008)). The importance of the initial centre selection is reflected in the possibility of generating different clusterings, each representing a local optima unless steps are taken to avoid this or the data is well-conditioned. In addition, the iterative nature of the algorithm means that an increase in the size of the data set greatly increases the time necessary to locate the local optimum (Zhang et al. (2008)).
2.2.3 Model based clustering

Mechanics

Distribution-based clustering works under the assumption that clusters originate from the noise around a true signal at the cluster’s centre, in other words, that the members of a cluster are sampled from the same probability distribution (Moon (1996)). It produces maximum-likelihood estimates of the set of parameter values that generate each of the clusters (Moon (1996)).

Example Algorithms

- Expectation-Maximization Algorithm

  - Similar to k-means clustering, Expectation Maximization involves the iterative application of two distinct steps: the estimation set and the maximization step (Moon (1996)). During the estimation step unobserved data is estimated using the current (k) parameter then in the maximization step the estimated data is used to calculate the maximum likelihood estimate of the next parameter (k+1) (Moon (1996)). The algorithm has found a local optimum when k and k+1 converge (Moon (1996)).

Benefits

Expectation-Maximization is ideal for dealing with a situation where measurements are not direct or some of the data is missing e.g. if data has measurements that are ranges
or if data collection is in some manner truncated (Moon (1996)). An added benefit of Expectation-Maximization is that due to the underlying use of probability functions to generate the clusters, each data point has a level of certainty associated with its assignment to a particular cluster (Fraley and Raftery (1998)).

**Drawbacks**

There are several drawbacks to Expectation-Maximization. Depending on the data set and the initial parameter values, convergence to a local optimum can be quite slow although if the data has distinct clusters and the initializing values are selected appropriately this is not usually a problem (Fraley and Raftery (1998)). Due to the nature of the algorithm, large numbers of features are needed to be considered to generate sufficiently distinct clusters. Expectation-Maximization cannot differentiate between clusters (Fraley and Raftery (1998)). Additionally, the algorithm requires the component covariance matrices to be properly conditioned i.e. not singular or nearly singular (Fraley and Raftery (1998)).

### 2.2.4 Density-based clustering

**Mechanics**

Density-based clustering methods determine cluster outlines based on data density; clusters are areas of high density whose boundaries are comprised of areas of low density (Agrawal et al. (1998)). The density around a particular point is calculated by the number of other points in its neighbourhood and a cut-off value is used to determine what
is considered a dense location (Sander et al. (1998)).

**Example Algorithms**

- **DBSCAN**

  - DBSCAN scans the database searching for points that are in the cores of the underlying clusters (Ester et al. (1996)). A point is a core point if the number of points in its Esp-neighbourhood exceeds a predetermined cut-off value (Ester et al. (1996)). If the point is a border point it is classified as belonging to the cluster of core points that are in its neighbourhood (Ester et al. (1996)).

- **GDBSCAN**

  - GDBCAN is a generalized version of DBSCAN. Instead of density-based clusters the algorithm uses density-connected sets (Sander et al. (1998)). The difference between the two is a simple alteration to the definition of neighbourhood as DBSCAN uses Eps-nieghbourhood and GDBCAN uses a binary predicate to define a point’s neighbourhood (Sander et al. (1998)).

**Benefits**

At the time of its creation, it offered a significant improvement in run-time (Ester et al. (1996)). DBSCAN has a good ability to cluster noisy data and for example out-performs k-means in detecting swallow events and offers a decrease in run-time and an increase in
consistency between patients compared to the quadratic based signal detection algorithm generally used for that function (Dudik et al. (2015)).

**Drawbacks**

DBSCAN is limited to data with 10 or fewer dimensions (Agrawal et al. (1998)) which make it unsuitable for many modern data sets. When it is applied to databases that contain data that fits its requirements it takes a lot of processing time to generate its classifications (Sander et al. (1998)).

### 2.2.5 Subspace clustering

**Mechanics**

Subspace clustering is an alternative to feature selection that attempts to find clusters in high dimensional data and note the features that are important for cluster definition (Parsons et al. (2004)). Clusters have the potential to overlap but are separated by their relevant dimensions (Parsons et al. (2004))

**Example Algorithms**

Some examples of this type of algorithm are:

- **CLIQUE**
  
  - CLIQUE is a bottom-up subspace clustering method that uses a static grid placed in the data space to identify which bins contain tight groupings of the
data (citeagrawal1998automatic, Parsons et al. (2004)). After these areas are identified, clusters are defined so that the fewest number of clusters are required to fit the observed patterns of density; redundant and disjoint clusters are pruned (Agrawal et al. (1998), Parsons et al. (2004)).

There are several offshoots of CLIQUE that attempt to improve upon the original algorithm. ENCLUS does not measure density directly and instead measures entropy as an indirect method for identifying clusters (Parsons et al. (2004)). MAFIA uses an adaptive grid instead of a static one for the definition of bin outlines afterwards following the same CLIQUE methodology (Parsons et al. (2004)).

- ORCLUS

- ORCLUS is derived from the first top-down subspace clustering algorithm and refines it by looking for subspaces that are not parallel to any axis (Parsons et al. (2004)). Points are assigned to cluster centres, the directionality of the spread of the cluster is determined, and clusters that are close to each other and spread in the same general direction are merged (Aggarwal and Yu (2000); Parsons et al. (2004)).

- COSA

- COSA examines each instance individually and iteratively updates the weighting of dimensions so that the dimensions that connect the instance to its k-
nearest neighbours are weighted more heavily (Parsons et al. (2004)). The weighting of each dimension is updated each iteration so that more value is assigned to the dimensions that have the smallest spread within the k-nearest neighbour group (Parsons et al. (2004)). The algorithm is complete when the weights remain constant between the nth run and the n-1th run (Parsons et al. (2004)).

- CLTree
  - CLTree uses a modified decision tree to generate clusters in a two-step process (Liu et al. (2000); Parsons et al. (2004)). First, the decision tree is built, followed by pruning based on the user input values (Liu et al. (2000); Parsons et al. (2004)). To build the tree, each dimension is examined and the location for the most productive cuts to separate high and low-density areas are found and implemented (Liu et al. (2000)).

**Benefits**

The algorithms in this category are widely varied in their approaches and therefore all have different strengths and weaknesses. CLIQUE scales well with increases in a number of data points in a data set (Agrawal et al. (1998)). CLTree, although it has a high initial computational cost, does scale well with increases in the number of dimensions and in the size of the data set (Parsons et al. (2004)).
Drawbacks

Some algorithms in this category eg. BIRCH require the user to input the number of clusters (Agrawal et al. (1998)) this is potentially problematic if there is a lack of previous research from which to draw the appropriate number of clusters. CLIQUE does not cope well with higher dimensional clusters but for most applications, this is generally not an issue (Parsons et al. (2004)). In addition CLIQUE and other clustering methods that use bins have difficulty scaling as the number of dimensions increase as this massively increases the number of bins that need to be evaluated (Parsons et al. (2004)). CLTree has two user input values that are responsible for pruning the tree and the resultant clusters are highly dependent on these values (Parsons et al. (2004)).

2.2.6 Where does bubble clustering fall?

Bubble clustering defines cluster boundaries and the closeness between points by a repeated measure of association as an attempt to approximate the underlying true association. These methods of defining clusters and their borders are compatible with the connectivity-based clustering definitions and therefore bubble clustering falls into this classification.
2.3 Evaluating Clustering Results

2.3.1 Calculating the Distance Between Two Hierarchical Trees

To be able to evaluate the stability of a clustering, we first need a way of telling how
distant two trees are from each other. The distance between two trees with an identical
set of leaves can be obtained by computing the distance between the vector of minimum
containing clade (MCC) (Definition 1) size for all pairs of points for both trees (Ashlock
et al. (2009)) . For example, the size of the MCC for \{2,4\} in Figure 2.1 is 3 as the
smallest possible clade that contains both 2 and 4 also contains 6 and the MCC for \{1,6\}
is 6 because the entire tree is needed to obtain a clade that contains both 1 and 6. The
MCC-vector contains one entry for each pair of taxa. The value of this entry is the number
of leaves in the MCC for that pair of taxa. The MCC-vector allows the tree to be mapped
in a 1-1 fashion - two trees only share the same MCC – position in the data space – if they
share the same labeled topology (Ashlock et al. (2009)) . Due to this 1-1 mapping there is
no need to normalize tree presentation prior to assessing the distance between them.

Definition 1 The minimal containing clade for two taxa is the smallest sub-tree in which
both appear. Its size is the number of leaves within that minimal sub-tree.

The distance between two vectors given by Formula 2.1.

If \( \vec{u} = (u_1, u_2, \ldots, u_n) \) and \( \vec{v} = (v_1, v_2, \ldots, v_n) \) then the Euclidean distance is given
by:

\[ d(\bar{u}, \bar{v}) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \]  

(2.1)

The distance between two trees that are identical in topology is 0 as in the case of Figures 2.2, 2.5 and 2.7 because all the values for the MCC for all possible pairs of points will be the same. This is completely independent of how the tree is represented graphically, clades can be rotated without disrupting the internal structure. The more different the two trees are the higher the distance between the two as two points that were close together become farther away resulting in one of the MCC values becoming substantially different from the equivalent MCC in the other tree.

2.3.2 The Stability Measure

The stability measure used in this study, Ashlock et al.’s (Ashlock et al. (2009)) method, is based on calculating the difference between trees constructed with the full data set with the leaf of a data point simply removed (the snip tree) and those generated with a the focal data point removed (the rebuilt tree). The distance between trees is then summed over all possible snip and rebuild tree pairs to obtain the total stability and averaged to get the average stability per Snip-Rebuild pair. The method for determining the distance between two hierarchical trees is explained in detail in Section 2.3.1.

For example, the constructed data set in Table 2.1 can be used to build the dendrogram in Figure 2.1 when all of the data is run through the hclust() algorithm. To generate the snip trees needed to calculate the stability, individual leaves are simply removed (Figure
Table 2.1: Constructed data set used to illustrate Ashlock et al.’s (Ashlock et al. (2009)) method for testing stability

<table>
<thead>
<tr>
<th></th>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Feature 3</th>
<th>Feature 4</th>
<th>Feature 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>0.4120052</td>
<td>1.020401</td>
<td>1.621813</td>
<td>-2.6195337</td>
<td>-3.7227263</td>
</tr>
<tr>
<td>Sample 2</td>
<td>2.5459116</td>
<td>-4.545647</td>
<td>-2.419535</td>
<td>-1.5746646</td>
<td>0.4149181</td>
</tr>
<tr>
<td>Sample 3</td>
<td>-1.9172850</td>
<td>1.466670</td>
<td>4.468418</td>
<td>0.9450918</td>
<td>0.1504432</td>
</tr>
<tr>
<td>Sample 4</td>
<td>-1.5183032</td>
<td>3.439790</td>
<td>-1.965946</td>
<td>-3.2277932</td>
<td>3.2973363</td>
</tr>
<tr>
<td>Sample 5</td>
<td>0.6086980</td>
<td>-2.947627</td>
<td>3.893053</td>
<td>0.8533524</td>
<td>0.8085557</td>
</tr>
<tr>
<td>Sample 6</td>
<td>-2.4414115</td>
<td>-3.412438</td>
<td>-1.791040</td>
<td>0.3749243</td>
<td>-1.9199126</td>
</tr>
</tbody>
</table>

1.2). These then need to be compared to the rebuilt trees (Figures 2.2 - 2.7). There are three different outcomes that can happen when comparing the snip and rebuild trees: the trees are identical in topology and presentation (Figure 2.2), they are identical in topology but are presented differently (Figures 2.5, 2.7), or the trees are different (Figures 2.3, 2.4, 2.6).

To obtain the total stability of the complete tree the distances between all the snip and rebuild trees need to be summed together. The average stability per Snip-Rebuild pair is the total stability divided by the number of points in the data set. An example of the snip and rebuild MCC vectors can be found in Table 2.2. From these, the distance between the trees can be measured. In this example, the distance between the two is:

\[ \sqrt{0^2 + 0^2 + 2^2 + 0^2 + 0^2 + (-2)^2 + 0^2 + 2^2 + 0^2 + (-2)^2} \]

\[ = \sqrt{4 + 4 + 4 + 4} \]
Figure 2.1: Tree using hclust() using the data in Table 2.1
Figure 2.2: Both trees represent dendrograms of the data in Table 2.1 with the 1st sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Figure 2.3: Both trees represent dendrograms of the data in Table 2.1 with the 2nd sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Figure 2.4: Both trees represent dendrograms of the data in Table 2.1 with the 3rd sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Figure 2.5: Both trees represent dendrograms of the data in Table 2.1 with the 4th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Figure 2.6: Both trees represent dendrograms of the data in Table 2.1 with the 5th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Figure 2.7: Both trees represent dendrograms of the data in Table 2.1 with the 6th sample removed. The snipped tree is on the top and the rebuilt tree is on the bottom.
Table 2.2: MCCs for each pair of points in the Snip and Rebuild trees with the 5th sample removed

<table>
<thead>
<tr>
<th></th>
<th>{1,2}</th>
<th>{1,3}</th>
<th>{1,4}</th>
<th>{1,6}</th>
<th>{2,3}</th>
<th>{2,4}</th>
<th>{2,6}</th>
<th>{3,4}</th>
<th>{3,6}</th>
<th>{4,6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snip</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Rebuild</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

= 4

The two remaining trees that are different between the snip and rebuild versions (Figures 2.3 and 2.4) have a stability measure of $\sqrt{22}$. The total stability for the hclust() generated tree for the data set in Table 2.1 is 13.38 and the average stability per Snip-Rebuild pair is 2.23. The greater the distance between the Snip and Rebuild pairs the greater the stability measure and the average stability per Snip-Rebuild pair.

2.3.3 The Rand Index as a measure of agreement between clusterings

The Rand Index and the Adjusted Rand Index are mathematical ways of determining agreement between two partitionings (Rand (1971); Yeung and Ruzzo (2001)). The Rand Index can take on a value between 0 and 1 with higher levels of agreement resulting in values closer to 1 which represents perfect agreement (Rand (1971); Yeung and Ruzzo (2001)). When comparing two clusterings of the same data set there are three possible states when comparing pairs of points. The points are in the same cluster in both clusterings, they are in different clusters in both clusterings or the two clusterings do not agree. The Rand Index is the fraction of pairs of points for which both clusters agree that they are either in
the same cluster or different clusters (Equation 2.2) (Rand (1971)).

\[ \frac{Agreement}{Agreement + Disagreement} \]  

(2.2)

2.3.4 Connection between the Stability measure for hierarchical trees and the Rand Index

The Rand Index determines the closeness of two clusterings by looking at agreement and disagreement. This results in a binary state, either a score of 0 or 1. Like the Rand Index, the tree distance measure assigns values to assess agreement between pairs of points; the differences are that 0 is used to denote agreement and that the measure of disagreement is given a range of weights from 1 to the number of leaves less two depending on the degree of change in the size of the MCC. This allows the tree distance measure to both fill the function of a Rand Index generalized so that it can be applied to hierarchical clusterings but also that it can contain more complex information about the degree of disagreement between the clusterings. It should be noted that while the Rand Index value increases as the level of similarity between two clusterings increases, the opposite is true for tree distance: the greater the distance the more dissimilar the two trees. In effect they are measuring the same quality but in opposite directions.
Chapter 3

Experimental Design

3.1 Introduction

In order to address the previously identified instability in the results produced by the standard hierarchical clustering method, a new hierarchical clustering algorithm - *bubble clustering* - is proposed. The algorithm functions by building up a matrix of association between data points by rewarding connections between points found to be in the same sphere placed into the data set. The spheres are centered on a randomly selected data point and the radius is randomly selected from the range of the minimum distance between points and the maximum distance between points. Preliminary results suggest that this new algorithm offers a significant improvement in result stability when compared to the standard methods.
3.2 The bubble clustering algorithm

Bubble clustering’s method to generate hierarchical trees can be broken into two distinct phases: the building of the *associator matrix*, and the generation of a hierarchical tree from the associator matrix. The associator matrix is initially created with all cells filled with the same very small, near zero value. This is so that if any pairs of points are never found in the same bubble the association between those points is not exactly zero to avoid calculation errors down the line. The associator matrix is completed by sequentially rewarding points that fall within the same multidimensional bubble. The steps for generating the *associator matrix* are listed in Algorithm 2.

**Algorithm 2 Bubble clustering - construction of associator matrix**

**Input:** Numeric data matrix with n samples  
*degree of bubble coverage*(L)

**Output:** Associator matrix

**Details:**

Initialize n x n associator matrix with 0.0000000001s

Repeat

*Select random center (c) from n samples*

*Generate random radius (r)*

*Number in bubble (N) set to 0*
For \( i \) in \( n \)

\[ \text{IF } \text{dist}(i, c) < r \text{ Increment N by 1} \]

For \( i \) in \( n \)

For \( j \) in \( n \)

\[ \text{IF } \text{dist}(i, c) < r \text{ AND } \text{dist}(j, c) < r \text{ associator matrix } (i,j) += 1/N \]

\[ \text{Until 10}^L n \text{ iterations have been performed} \]

Return: associator matrix

The R function version of bubble clustering allows the user to select the desired coverage used for associator matrix generation. The number of bubbles used to construct the associator matrix is \( 10^a n \) where \( a \) is the user input value and \( n \) is the size of the data set.

Once the associator matrix is complete the tree is then constructed by sequentially linking the two most associated data points that are not currently in the same sub-tree. For example, given the example associator matrix in table, 3.1 the first two points that would be joined are B and C, followed by A and D being joined. Next, the subtrees containing B and D are connected. This new sub-tree is then connected to E to make a complete tree (see Figure 3.1)

At their core, both hierarchical clustering using the neighbor joining algorithm and bubble clustering both iteratively join the most similar objects until the entire data set is
Table 3.1: Example associator matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>-</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>6</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 3.1: Tree built from the example *associator matrix* in table 3.1

connected. The difference lies in how the similarity is measured. Neighbour joining uses a single measure, Euclidean distance, as a way of determining "closeness", whereas bubble clustering uses a measure of association to determine "closeness" that is built through repeated measures. Because the repeated measures involve multiple points, the effect of any one particular point on the level of association between other points is greatly reduced and therefore increases the stability of the results in the face of the addition or removal of a point. This hypothesis is tested in the experimental section.
3.3 Comparing bubble clustering and hclust() outputs with different numbers of dimensions

Several different experiments were performed. Data sets were generated by sampling enough numbers from the random uniform distribution between -5 and 5 to fill various sized matrices. Three different matrix sizes were examined: 50x2, 50x5, 50x10. For each of the experiments once a data set was generated the stability of that data set was calculated using three different tree-building methods: bubble clustering with 10n bubbles where n is the number of points, Bubble clustering with 100n bubbles where n is the number of points, Bubble clustering with 1000n bubbles where n is the number of points, hclust() using the complete linkage method, hclust() using the average method and hclust() using the centroid method. Evaluations of stability were performed on stability information of 100 runs of each experiment.

Stability was measured using Ashlock et al.’s (Ashlock et al. (2009)) method. Figure 3.2 shows a snip and a rebuilt tree generated using hclust() on random data. The distance between these two trees can be calculated by using Formula 2.1. For this example, the distance between the two trees is 16.4924225. The stability of the generated tree is the sum of the distances between the snip and rebuilt trees for all data points. The average stability per point is simply this result divided by the number of points.

Algorithm 3 Stability test of random data
Figure 3.2: Examples of a Snip and a Rebuild tree generated using hclust(method=complete) on random data

**Input:** Desired number of runs $n$

- size of desired matrix $(i,j)$
- original data matrix

**Output:** Stability scores

**Details:**

Repeat

- Generate $i \times j$ numbers selected from the random uniform distribution between -5 and 5
- Fill the $i \times j$ matrix with the generated numbers
- For $n$ in $i$
  - generate Rebuild tree with all data except sample $n$
calculate the MCC vector for Snip tree from original tree data

calculate the MCC vector for Rebuild tree from rebuilt tree data

calculate distance between Snip and Rebuild vectors

add distance to the growing total stability measure

Divide total stability by number of points in data set to get the average stability per Snip-Rebuild pair

Store results

Until desired number of runs complete

Return: Stored results

3.4 Evaluating the effect of data set size on stability

To evaluate the effect of data set size on the stability of the produced trees, transformation is required to make comparisons. This is because the more points that a tree has the more pairs of points that need to be examined to measure the distance between trees. There is also an increased likelihood of at least one point not being in exactly the same location within the tree as there are more points. To be able to compare between data sets the results need to be transformed to all be within the same range. So that the results could be compared they were scaled to fit within the range of 0-1. To accomplish this, the results for each of the 100 runs with each type of data set were transformed using the formula
in Equation 3.1. Where $R$ is the individual tree stability calculation, $Min$ is the minimum stability for all runs with the same number of data points and $Max$ is the maximum stability for all the runs with the same number of data points.

$$NormalizedValue = \frac{R - Min}{Max - Min}$$

(3.1)

3.5 Testing the stability of a real data set -

problem taxonomy

The data set is a collection of normalized times to solve optimization problems with fifteen different optimization algorithms. All of these algorithms are geographically structured evolutionary algorithms Bryden et al. (2006) with each using a different geography. The problems that the algorithms are tasked with solving are the One Max problem, three different plus-one-recall-store problems, ten real parameter optimization problems from Kenneth DeJong’s test suite, seven instances of the self-avoiding walk problem, the (6,3)-DNA barcode problem, six combinatorial design location problems, sorting lists of length 8, 9, and 10 points, maximizing the period of permutations on 30, 32, 34, or 36 points, and four instances of the parity problem in logic gate induction Ashlock (2006).

Each algorithm is applied to each problem and the number of fitness evaluations needed to solve a problem – averaged over 1000 trials – are normalized so that the range from fastest to slowest algorithm is mapped to the interval [0,1]. This removes information on
the difficulty of the task while preserving the relative speed of problem solution on different geographies. A hierarchical clustering of this information creates a taxonomy of problems that help estimate what type of algorithm is best used to solve a given type of problem.

The Problem Taxonomy data set was tested for the stability of the resultant hierarchical trees generated using bubble clustering with $10^a n$ where $a=1,2,3$ and the size of the data set was 34 and the previously tested methods of hclust() - complete, average and centroid. Ten runs were performed to examine the variability in stability. A second test of stability was performed with 15 runs of $a=3,4,5$ and the three hclust() methods.

The actual hierarchical trees generated by the two algorithms were also evaluated and the distance between the hclust(method=complete) and the trees generated by bubble clustering with $a=4$ and $a=5$ compared.

To test to see if the internal structure of the data does affect the stability of the trees constructed using the problem taxonomy data set, the stability of trees on randomized data sets pulled from the same data space were tested. To accomplish this the scores for each problem were re-ordered independently in R using the shuffle() function. The stability of the trees generated from the newly generated data set was evaluated and recorded for bubble clustering with $10^a n$ bubbles where $a=1,2,3$ and the three hclust() methods. This process was repeated 50 times to ensure a good sampling of the data space. Welch’s t-tests were then performed to determine if the results between the real data and shuffled data tests were significantly different.
3.6 Testing the stability of a real data set - DNA dimers

For a biological data set records consisting of the frequencies of DNA dimers in primers was selected. The primers were developed for an EST project in corn at Iowa State University (Emrich et al. (2004)) Each data set contains 50 records with sixteen coordinates corresponding to the the counts within the primer for sixteen possible dimers, CC, CG, ..., TT. These sequences were selected from 4954 available records and are intended to check the stability results on data with a biological origin.

The general stability of the three data sets was tested for bubble clustering with $10^a n$ bubbles where $a=1,2,3,4,5$ and the three hclust() methods. For the first of the three data sets 32 runs were performed and the other two data sets were tested with 15 runs each. The number of runs was reduced for the subsequent data sets, because the differences in stability measure between bubble clustering and hclust() on the first data set were sufficiently large that the minor variability in bubble clustering was not a concern.

To further examine the difference between bubble clustering trees and hclust() trees, examples of Snip-Rebuild pairs were constructed and the distance between the bubble clustering where $a=5$ results and the hclust(method=complete) was calculated.

3.7 Technical challenges

R is not a fast computing language, so the majority of the algorithm needs to be written in a compiled language. In this work the core routines were written in C++ and then
called from R. The main difficulty with this is that there is a great lack of appropriate error messages to indicate if the implemented code is behaving appropriately. The C++ code that performs most of the algorithm’s tasks is broken down into smaller functions. The three main functions are: generate the associator matrix, turn the information in the associator matrix into the information required for R to generate a dendrogram, and the functions to calculate the MCC of both the snip and rebuild trees.

Given the nature of the stability measure examining the distance between trees considering every possible pairing of data points, direct comparison between stability values of differently sized data sets are not possible as the number of possible data point pairs are not the same.
Chapter 4

Results

Bubble clustering shows improved stability when compared to all three of the examined hclust() variants. The since the stability measure is the distance between snip and rebuild trees, lower values are better. For all of the figures in this section the algorithms are labeled using the following scheme: all of the bubble clustering algorithms are B1, B2, B3, ..., Bn where the number denotes the level of bubble coverage used to construct the associator matrix e.g. B1 uses \(10^1n\) bubbles, B2 uses \(10^2n\) etc., the hclust() algorithms tested are labeled in a similar way with H1 representing hclust(method=complete), H2 representing hclust(method=average) and H3 representing hclust(method=centroid). Boxplots that share a letter are not statistically significantly different \((p \geq 0.05)\) while those that have different letters are statistically significantly different \((p < 0.05)\).
4.1 Effect of different number of dimensions on the stability measure

In data sets with 50 data points, the average instability of trees generated with all tested number of bubbles was less \((p < 0.05)\) than all hclust() methods tested when the data was 2, 5 or 10 dimensional (Figures 4.1, 4.2, 4.3). Each of the exponential increases in the number of bubbles used to build the associator matrix generated a significant \((p < 0.05)\) improvement in the stability. Trees made with \(10^1 \times 50\) bubbles were more accurate than those made by any of the hclust() variants but were less stable \((p < 0.05)\) than those made with either \(10^2 \times 50\) bubbles or \(10^3 \times 50\) bubbles. In turn, the trees made with \(10^3 \times 50\) bubbles were significantly \((p < 0.05)\) more stable than those made with \(10^2 \times 50\) bubbles. Increasing the number of bubbles further continued this trend and \(1000 \times 50\) bubble produced significantly better result than \(100 \times 50\) bubbles. Among the different hclust() methods tested complete linkage produced the most stable trees \((p < 0.05)\). While centroid and average hclust() methods produced similar results \((p \geq 0.05)\) with two-dimensional data sets centroid performed significantly worse \((p < 0.05)\) when the number of dimensions was increased to five and continued to be worse with ten-dimensional data sets.

When comparing the level of instability with the algorithm remaining the same but the dimensions changing an interesting pattern emerges. For all examined bubble clustering variants there is a significant improvement \((p < 0.05)\) in the average stability measure as the
Figure 4.1: Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 2 observations randomly generated between -5 and 5. Label description in the text.

Figure 4.2: Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 5 observations randomly generated between -5 and 5. Label description in the text.
Figure 4.3: Average stability per Snip-Rebuild pair results on randomly generated data sets with n=50 data points each with 10 observations randomly generated between -5 and 5. Label description in the text.

dimensions in the data increase (Figures 4.4, 4.5, 4.6). Hclust() methods examined however show a significant ($p < 0.05$) decrease in the average stability as the dimensionality of the data increases (Figures 4.7, 4.8, 4.9).

4.2 Effect of size of data set on stability of trees

Random data sets with values between -5 and 5 were constructed to have 25, 50 or 100 points in $\mathbb{R}^{10}$. As the number of comparisons made to generate the stability statistic increases non-linearly as the size of the data set increases the stability scores of the trees
Figure 4.4: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 10x50 bubbles (B1).

Figure 4.5: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 100x50 bubbles (B2).
Figure 4.6: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by bubble clustering with 1000x50 bubbles (B3).

Figure 4.7: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the complete linkage method (H1).
Figure 4.8: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the average method (H2).

Figure 4.9: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with 2, 5 and 10 observations randomly generated between -5 and 5 as produced by hclust() using the centroid method (H3).
produced were standardized to be between 0 and 1 with the lowest score of a tree made with n points set to 0 and the highest score set to 1. For all tested versions of bubble clustering, there was a significant \( p < 0.05 \) improvement in stability when the number of data points was increased from 25 to 100. For trees made with \( 10^1n \) bubbles, there was a significant \( p < 0.05 \) improvement when the data set increased from 25 to 50 points and from 50 points to 25 points. However, when the number of bubbles used to construct the associator matrix was \( 10^2n \) or \( 10^3n \) bubbles while there was a significant \( p < 0.05 \) increase from 25 to 50 points the increase in average stability was not significant \( p \geq 0.05 \) when the data set size was increased from 50 points to 100 points. The relationship between stability and data set size was the opposite for hclust(method=complete) and hclust(method=average) variants with both getting significantly \( p < 0.05 \) more unstable as the number of points increased. For hclust(method=complete) the difference arises between 25 to 50 but does not get significantly \( p \geq 0.05 \) worse between 50 and 100 points. hclust(method=average) follows the same pattern as hclust(method=complete) either the average stability of trees with 50 leaves being worse than those with 25 leaves but not statistically different from those with 100 leaves. There is no statistically significant difference between the stability scores obtained in any of the hclust(method=centroid).
Figure 4.10: Average stability per Snip-Rebuild pair results on randomly generated data sets with 50 data points each with \( n = 25, 50 \) and 100 data points observations randomly generated between -5 and 5. To allow comparison between different sized data sets resultant stability per Snip-Rebuild pair was normalized so that the results of the 25, 50 and 100 runs all individually fit in the range of \([0,1]\).
Figure 4.11: Average stability per Snip-Rebuild pair results on the Problem Taxonomy data set n=34 with 10 runs each

Table 4.1: Variance of Stability Scores of bubble clustering trees in Figures 4.11 and 4.12

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>B1 n=10</th>
<th>B2 n=10</th>
<th>B3 n=25</th>
<th>B4 n=15</th>
<th>B5 n=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability Score Variance</td>
<td>253.31</td>
<td>322.46</td>
<td>33.12</td>
<td>3.87</td>
<td>0.49</td>
</tr>
</tbody>
</table>

4.3 Testing the stability of a problem taxonomy data set

The first test of the stability of the hierarchical trees indicates that there is no variation in the stability of hclust() hierarchical trees given a single data set, every time the calculation is run the same stability score is obtained (Figure 4.11). This is not the case for trees constructed using bubble clustering (Figure 4.11) although, the variance tends to decreases as the number of bubbles used increases (Table 4.1). The one exception to this trend is B2 although when the possible outlier is excluded the variance of the remaining 9 runs is 65.22 which does fit the pattern. A second test was conducted to get a better idea of the number
Figure 4.12: Average stability per Snip-Rebuild pair results on the Problem Taxonomy data set n=34 with 15 runs each)

of bubbles required to reduce the variance to an acceptable limit (Figure 4.12).

After it was established that both $10^4 n$ bubbles and $10^5 n$ bubbles generated relatively low variance the trees built using the whole data set were examined. As shown in Figure 4.13 two separate runs of $10^4 n$ bubbles generate trees with the same topology with slight variation in the heights of various branches. The same tree was also the result of $10^5 n$ bubbles (Figure 4.14). These trees are quite different from the tree found by hclust(method=complete) - the most stable version of the algorithm (Figure 4.15). Specifically, the distance between the bubble clustering tree and the hclust() tree is 287.3952 which is greater than the average distance between Snip-Rebuild pair for both bubble clustering $10^4 n$, bubble clustering $10^5 n$ or hclust(method=complete).

Trees for a random Snip-Rebuild pair were selected to illustrate the instability asso-
Figure 4.13: Two example trees of the problem taxonomy data set constructed using bubble clustering with $10^4 n$ bubbles.
Figure 4.14: Hierarchical tree of the problem taxonomy data set constructed using bubble clustering with $10^5 n$ bubbles.

Figure 4.15: Hierarchical tree of the problem taxonomy data set constructed using hclust(method=complete).
Figure 4.16: Snip hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with bubble clustering with $10^4 n$ bubbles.

associated with bubble clustering ($10^4 n$ bubbles) and with hclust(method=complete). The trees shown in Figures 4.16, 4.17, 4.18 and 4.19 show the differences between the Snip and Rebuild trees. The distance between the Snip and Rebuild trees with Sample 17 removed for bubble clustering was only 9.591663 whereas the distance between the two for hclust(method=complete) was 143.5131. The distance between the Rebuilt trees using the two different methods was 287.941.

To examine the effect of the true underlying structure on the stability results the observations in each column of the original data set was independently re-ordered and the stability of trees resulting from this data set was evaluated with $10^1 n$, $10^2 n$, $10^3 n$ bubbles bubble clustering and with hclust(method=complete), hclust(method=average), hclust(method=centroid). The data set was re-ordered 50 times and the stability recorded (Figure 4.20. All of the al-
Figure 4.17: Rebuilt hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with bubble clustering with $10^4n$ bubbles.

Figure 4.18: Snip hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with hclust(method=complete).
Figure 4.19: Rebuilt hierarchical tree of the problem taxonomy data set with Sample 17 removed constructed with hclust(method=complete).

Algorithms generated statistically different stability scores on the shuffled data with $10^3 n$ bubbles performing the best and hclust(method=centroid) performing the worst.

The stability scores of the shuffled data were then compared to their real data counterparts, i.e. hclust(method=complete) scores were compared to each other etc. and tested for significant differences (Figure 4.21). Most of the algorithms produced statistically different results when applied to the shuffled data when compared to the real data ($p < 0.05$) except hclust(method=centroid) for which the two were not significantly different ($p = 0.7881$). Of those that were statistically different, only hclust(method=complete) produced more stable trees when applied to the real data than those constructed with the average shuffled data set. All six algorithms have some overlap between the stability results between the stability scores on the real and shuffled data although for some the overlap is greater than
4.4 Testing the stability of a DNA dimers data set

The stability of the first DNA dimer data set - DimersA - included tests for the stability of bubble clustering with $10^n$ bubbles with $a=1,2,3,4$ and 5 and hclust(method=complete), hclust(method=average) and hclust(centroid). The test was allowed to run over-night and accomplished 32 runs (Figure 4.22). The other two data sets - DimersB and DimersC - were tested for stability with 15 runs each (Figures 4.23, and 4.24). For all runs bubble clusters with all levels of coverage out performed all hclust() variants on all three data sets. Bubble clustering performed similarly on all three data sets with the stability results falling into
Figure 4.21: Comparison of average stability of the real problem taxonomy data set and re-ordered versions
the same general areas. On the other hand, hclust() showed a different response based on the data set. For instance, hclust(method=complete) had a much higher stability score for DimersB compared to DimersA and DimersC. In addition, DimersB was the only data set for which hclust(method=average) had significantly lower stability scores ($p < 0.05$) than hclust(method=centroid) although the two remain the two with the highest scores. As with the problem taxonomy data set, bubble clustering with $a=4$ and with $a=5$ did not produce significantly different ($p \geq 0.05$) results when applied to any of the three dimer data sets (Figures 4.22, 4.23, and 4.24).

Once again, the trees produced by the two algorithms are very dissimilar. The hclust (method=complete) trees all have distinct clusters with deep divides whereas the bubble clustering trees show a more linear structure to the data (Figures 4.25-4.30). The distance
Figure 4.23: Average stability of Snip-Rebuild pairs generated from second of the DNA dimer data sets n= 50 with 15 runs

Figure 4.24: Average stability of Snip-Rebuild pairs generated from third of the DNA dimer data sets n= 50 with 15 runs
between the two trees constructed using dimersA is 564.2748. The distance between the bubble clustering tree and the hclust(method=complete) tree constructed from dimersB data is 677.0421. The distance between the two dimers C trees is 718.6849. When visually comparing the two interpretations of the data we can see that there are only a few data point relationships that are present in both. Only half of the six pairs that have an MCC of size 2 in the bubble clustering tree have an MCC of 2 in the hclust(method=complete) tree. The remaining three pairs have an MCCs of sizes 3, 5 and 12.

The difference in instability is detectable when examining Snip and Rebuild trees (Figures 4.31 - 4.42). The pairs with an MCC size of 2 in the full dimersB tree constructed with bubble clustering are retained when sample 10 is snipped (Figures 4.35-4.36).
Figure 4.26: Hierarchical tree of the dimersA data set constructed using `hclust(method=complete)`.

Figure 4.27: Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^5n$ bubbles.
Figure 4.28: Hierarchical tree of the dimersB data set constructed using hclust(method=complete).

Figure 4.29: Hierarchical tree of the dimersC data set constructed using bubble clustering with $10^5n$ bubbles.
hclust(method=complete) tree for dimersB also maintains all of its MCCs of size 2 but the membership of the larger subtrees is not consistent between the Snip and Rebuild trees. The Snip tree’s first division of the data into two clusters partitions the data into a sub-tree with 26 leaves and a sub-tree with 23 leaves whereas the Rebuild tree splits the data into a sub-tree with 21 leaves and another with 28 leaves (Figures 4.37-4.38). This change in the deep divide is reflected in the distance between the Snip and Rebuild trees. The distance between the two hclust(method=complete) trees for dimersB is 593.1155 when the distance between the two bubble clustering trees is only 24.65766. Similar values can be found when comparing the other two pairs of trees. The dimersA bubble trees are 35.29873 units apart while the hclust(method=complete) trees are 504.0357 units apart. The bubble and hclust(method=complete) trees for dimersC are, respectively, 59.58188 and 551.7209
Figure 4.31: Hierarchical tree of the dimersA data set constructed using bubble clustering with $10^5n$ bubbles with sample 48 snipped.

Figure 4.32: Hierarchical tree of the dimersA data set constructed using bubble clustering with $10^5n$ bubbles with sample 48 removed from the data set.
Figure 4.33: Hierarchical tree of the dimersA data set constructed using hclust(method=complete) with sample 48 snipped.

units apart.
Figure 4.34: Hierarchical tree of the dimersA data set constructed using hclust(method=complete) with sample 48 removed from the data set.

Figure 4.35: Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^{5/n}$ bubbles with sample 10 snipped.
Figure 4.36: Hierarchical tree of the dimersB data set constructed using bubble clustering with $10^n$ bubbles with sample 10 removed from the data set.

Figure 4.37: Hierarchical tree of the dimersB data set constructed using hclust(method=complete) with sample 10 snipped.
Figure 4.38: Hierarchical tree of the dimersB data set constructed using hclust(method=complete) with sample 10 removed from the data set.

Figure 4.39: Hierarchical tree of the dimersC data set constructed using bubble clustering with $10^5/n$ bubbles with sample 3 snipped.
Figure 4.40: Hierarchical tree of the dimersC data set constructed using bubble clustering with \(10^5n\) bubbles with sample 3 removed from the data set.

Figure 4.41: Hierarchical tree of the dimersA data set constructed using \texttt{hclust}(method=complete) with sample 3 snipped.
Figure 4.42: Hierarchical tree of the dimersC data set constructed using hclust(method=complete) with sample 3 removed from the data set.
Chapter 5

Discussion

5.1 Effect of different number of dimensions on the stability measure

In all of the tests on the randomly generated data sets, bubble clustering offers a significant improvement in terms of stability when compared to the tested hclust() variants. While the stability of generated trees significantly decreased when hclust() variants were applied to random data sets when the dimensions of the data were increased from only two dimensions to ten dimensions, there was a significant improvement on the stability of trees generated with bubble clustering.
5.2 Effect of size of data set on the stability of trees

The stability of bubble clustering generated trees increased when the number of data points increased from 25 to 100 while the stability of hclust() generated trees tended to decrease as the number of points increased within the studied range. The exception to this trend was that the stability scores of trees made with hclust(method=centroid) did not get significantly worse as the number of data points increased although the average was close to the maximum observed stability measures and so differences may have been lost in the transformation necessary for comparison. It is also possible that the hclust(method=centroid) method is less susceptible to the instability generated by points positioned on the boundaries between more defined groups of points and that for most of the other tested algorithms the increase in the number of points increases the likelihood that there will be more of these problematic points. The addition of a problematic point can cause the centres of emerging clusters to shift in such a way that the clustering shift dramatically in a cascading fashion. There are multiple ways this could occur as the more points there are the higher the chance that at least a small number of them are problematic for clustering and the more points there are the smaller groups of points whose borders need to be considered and it is likely a combination of both that generate the end results.

It should also be noted that not all increases in data set size continued to have a significant effect. These together suggest that the relationship between the number of data points and resultant tree stability is more complex than a simple linear relationship or that the tested data set sizes were too close in size to get a complete picture of the relationship. It
should also be noted that all data sets were generated from the same data space with points
having coordinates between -5 and 5 regardless of data set size. It is possible that the data
space may interact with the number of data points when determining the stability of gen-
erated trees. Further study would be required to more accurately define how the number
of data points affects the stability of resultant trees especially with respect to the size and
shape of the data space.

5.3 Testing the stability of a problem taxonomy data set

Some of the trends observed in results of random data sets can be seen in the results of
the Problem Taxonomy data set. Bubble clustering tends to generate more stable trees the
more bubbles are used to build the associator matrix and the tested hclust variants follow
the same stability patterns as previously seen – hclust(method=complete) is more stable
than either hclust(method=centroid) or hclust(method=average). The increase in stability
when increasing the number of bubbles does have its limit though as $10^4n$ and $10^5n$ bubbles
did not have significantly different ($p \geq 0.05$) results. This would suggest that there is an
optimal number of bubbles needed to generate trees with the highest possible stability. For
this particular data set $10^4n$ bubbles are likely sufficient as there is relatively low variance
in the Stability Scores and produces trees with the high stability. More testing would be
required to determine the relationship between the optimal number of bubbles and the size
and type of the data set.
One of the differences seen in these tests was that the worst version of bubble clustering \((10^n \text{ bubbles})\) produced less stable results than \texttt{hclust(method=complete)}. While this is not an issue for bubble clustering as increasing the number of bubbles provided substantially better results it does suggest that the two algorithms are affected by the internal structure of the data set differently. Interestingly when compared to randomized data sets from the same data space the Problem Taxonomy data set is highly unstable for the examined levels of bubble clustering and \texttt{hclust(method=average)} but was more stable than the average random data set when \texttt{hclust(method=complete)} was applied and had an average instability when \texttt{hclust(method=centroid)} was applied. From this, we can infer that the inherent stability of the data set is dependent on the algorithm used to generate the clustering. While bubble clustering still performed better if found this clustering problem more difficult than other data sets in the same data space. \texttt{hclust(method=complete)} on the other hand found it an easier clustering problem than the average random data set but in the end still produced less stable trees.

### 5.4 Testing the stability of a DNA dimers data set

The test performed on the DNA dimers sets support the conclusions found in the other studies. Bubble clustering still produces more stable trees even when applied to biological data. Like the problem taxonomy data set there is no statistical difference \((p \geq 0.05)\) between the stability of bubble clustering with \(10^4 \times 50\) bubbles and bubble clustering with
$10^5 \ast 50$ bubbles. This further suggests that $10^4 \ast n$ bubbles might be a reasonable default value. `hclust(method=complete)` still outperforms the other two tested `hclust()` variants. Even though it is the most stable `hclust()` variant `hclust(method=complete)` still exhibits major differences between Snip and Rebuild trees. Unlike the problem taxonomy data set, $10^1 \ast 50$ bubbles were sufficient to generate more stable trees than `hclust(method=complete)`.

Interestingly `hclust(method=complete)` did not perform equally badly on all of the dimer data sets. The stability score for dimersB was much closer to the B1 values when compared to the results for the other two data sets. These two points together suggest that relative performance is affected by the data set. The dimersB data set also exhibited a reversal on which of `hclust(method=average)` and `hclust(method=centroid)` performed worse.

### 5.5 Algorithm search space

The two algorithms didn’t just differ in the stability of the results that they generated, they also produce very different trees. The distance between the bubble clustering tree and the `hclust(method=complete)` trees was larger than the average Snip-Rebuild distance for either algorithm regardless of data set among those tested. This suggests that the trees being produced by the algorithms occupy different locations within the possible tree space.

Each hierarchical clustering is a hypothesis about the underlying relationships present in the data. The stability of a particular tree reflects the size of the revisions to overall tree structure generated by the addition of individual points. This is related to the level
of confidence that can be attributed to any particular hypothesis of data relationships. If a model only changes slightly with the addition of additional information, the hypothesis it is more precise than one that majorly revises the relationships upon the addition of new data. In the particular case of evaluating the distances between Snip and Rebuild trees if there is a large distance between multiple Snip-Rebuild pairs, there is a large probability that the hypothesis is not close to the underlying truth except by chance. In the example of the problem taxonomy data, the two trees the position of only a single point moved, but was within the same clade with nine leaves, between the Snip and Rebuild trees constructed (Figures 4.16 and 4.17). The Snip-Rebuild trees for hclust(method=complete) (Figures 4.18 and 4.19) on the other hand majorly revise the location of several points. These trends continue for the dimers data sets as for the bubble clustering Snip-Rebuild pairs only a couple of points changed position whereas the membership of the two major clades in Figure 4.38 is not the same as in Figure 4.37.

5.6 Conclusion

Bubble clustering offers a significant increase in tree stability when compared to previous methods. Even in the most sparse coverage of the data space examined (10n bubbles placed to generate the associator matrix), improves upon the hclust outputs in all randomly generated data cases. More bubbles appear to be needed to offer a significant improvement in stability when the algorithms are applied to some real data, although the increasing
the number of bubbles used to generate the associator matrix quickly improves the sta-

dibility beyond the capabilities of hclust(). While the current version of bubble clustering
is better than the hclust() variants it is likely, not optimized to generate the highest possi-
ble stability. Further improvements may be possible after a more extensive parameter
study has been performed. Another pertinent result is that not all hclust() methods perform
the same on the tested random data sets or on real data sets. All of the tests demonstrate
the trend that hclust(method=complete) - which is the default- performed better than both
hclust(method=average) and hclust(method=centroid). Which of hclust(method=average)
and hclust(method=centroid) performed the worst depend on data set.

The most pertinent finding of the performed tests is that some data set qualities that
tend to generate poor stability in hclust() built trees instead increase the stability of trees
generated by bubble clustering. While it remains to be tested to see if the trends hold for
even larger and more complex data sets than those tested here bubble clustering shows
great promise as a hierarchical clustering algorithm. Its performance appears to improve
with larger data sets and with an increase in dimensionality which is the exact opposite of
the algorithms used for comparison. In addition, it seems to search a completely different
area of tree space and generates hierarchical trees that offer a very different hypothesis on
the underlying connections between the points than those suggested by hclust() trees. This
is potentially due to the differences in tree construction. Bubble clustering measures the
association between points and builds the tree based on individual associations. If a point’s
closest association is already in a cluster that point is simply added to the already existing
cluster. Neighbor joining, on the other hand, condenses the location of a cluster to a single point through methods that vary depending on the variant. This likely results in a distortion of the space of data point connections and it affects the resultant hierarchical tree. Points that may be close together may appear further apart to the algorithm if one of the two is already in a cluster. This biases neighbour joining towards starting new clusters instead of adding points to previously created clusters.
Chapter 6

Future directions

Given that bubble clustering is a completely new method for constructing hierarchical trees, there are a large number of additional steps required to fully understand the improvements observed in this study. The future directions for examination and characterizations of the behaviour of bubble clustering and the methodology at its core can be broadly divided into three categories: investigation into the relationships between the various parameters and the resultant stability, broadening the tests on biological and non-biological data to include more varied information sources, and examine the efficacy of bubble clustering and neighbor joining on quantifiably different types of data to determine the optimal applications of the algorithm.
6.1 Exploration of algorithm details

It is unknown if the reward of $1/N$ for being in the same bubble is the ideal reward structure. It might be better to have a reward that is less for points on the outside of the bubble and more for those closer to the centre. The could either be done by multiplying the reward by the ratio of the distance from the centre and the bubble’s radius or there could be set levels within the bubble. Alternatively, it may work just as well for the same reward each time.

Shared bubbles are not the only association metric that can be used to generate the associator matrix. Multiple point packings (Ashlock and Graether (2016)) placed into the data space might also be used to generate an association metric. Data points are associated if they are closest to the same point packing point. It would also be possible to generate Voronoi tilings (Aurenhammer (1991)) of the data space by randomly selecting a small subset of points. Each time a Voronoi tiling of the data space is generated points within the same tile gain an association reward.

The concepts behind bubble clustering can also be applied to non-matrix types of data, all that is required is a way to measure association. For example, DNA strings can be associated if they share the same motif, either exactly or with some leniency generated by the use of wildcards in the motif being used. This would only generate association in a present/absent manner. It is also possible to once the best match is found within a string to reward based on the distance between the DNA string and the motif by use of scores from a matrix containing the likelihood of substitutions. The previously proposed uses of
associator metrics would also be applicable to proteins instead of DNA strands.

6.2 Improvements to the stability measure

Currently the average Snip-Rebuild distance is calculated using all possible Snip-Rebuild pairs. Due to the nature of the MCC calculations this makes it not practical for data sets larger than 100 points. While the number of Snip-Rebuild pairs increases linearly the increase in the number of MCC calculations is not as the number of pairs of points increases at a much higher rate. A potential solution that need to be explored to increase the scalability of the Stability Measure could be to only sample the possible Snip-Rebuild pairs.

6.3 Applying bubble clustering to different data sets

Further testing also needs to be done to examine the stability of trees generated on a more diverse group of types of data sets. All the presented data was generated using completely random numbers selected from the uniform distribution between -5 and 5, some preliminary investigation of the problem taxonomy data set and the DNA dimer data sets. It would be of use to generate data sets that have different structures. Some examples of potential data sets are: data with some internal structure e.g. three distinct clusters, data from a more linear data space, data sets with more than 10 dimensions, data sets with experimental data instead of generated data.
6.4 Examination of behaviour on problematic data sets

This is an area that would benefit from evolutionary computing. One approach would be to evolve data sets that are hyper-unstable when neighbor joining is used to generate hierarchical trees and then examine the stability of trees generated using bubble clustering. This would allow us to partially test to see if the characteristics that contribute to instability in neighbor joining trees affect bubble clustering the same way. It would also perhaps be beneficial to examine the resultant data sets to see what data set characteristics are likely to affect the stability.

6.5 Further Applications of Stability Measure and Tree Distance Metric

There is a lot of untapped potential in the application of the Stability Measure and the Tree Distance Metric. Any clustering can be represented as a tree with all members of a non-hierarchical cluster connected to a single interior node and all interior nodes joined to form the top of the tree. This would allow for both a measure of cluster stability to be performed and it also permits the comparison of vastly different algorithms through a common generalized method. Hierarchical clusterings of data can also be turned into clusterings of k clusters by cutting the hierarchical tree at a height that generates k subtrees. This would permit comparison between hierarchical and non-hierarchical methods. The Stability Measure also has potential for determining the appropriate number of k by ex-
aming the changes in stability per Snip-Rebuild pair at different cut heights (hierarchical methods) or different user set value for k (non-hierarchical).
References


