Hierarchical Clustering and Tree Stability

Amanda Saunders and Daniel Ashlock and Sheridan Houghten

***Abstract*—Hierarchical clustering via neighbor joining, widely used in biology, can be quite sensitive to the addition or deletion of single taxa. In an earlier study it was found that neighbor joining trees on random data were commonly quite unstable in the sense that large re-arrangements of the tree occurred when the tree was reconstructed after the deletion of a single data point. In this study, we use an evolutionary algorithm to evolve extremely stable and unstable data sets for a standard neighbor-joining algorithm and then check the stability using a novel type of clustering called bubble clustering. Bubble clustering is an instance of associator clustering. The stability measure used is based on the size of the subtree containing each pair of taxa, a quantity that provides an objective measure of a given tree's hypothesis about the relatedness of taxa. It is shown experimentally that even in data sets evolved to be stable for a standard neighbor joining algorithm, bubble clustering is a significantly more stable algorithm.**

I. Introduction

Hierarchical clustering [6] places the elements of a data set at the leaves of a tree, providing a nuanced description of the relationships between data items. Items that are leaves of the same subtree are more closely associated with one another than those leaves not appearing in a subtree. The name *hierarchical* arises from the way that subtrees can contain their own subtrees. This study uses evolutionary computation to create data sets whose neighbor-joining trees exhibit exhibit exceptionally low and high levels of stability. Neighbor joining [8][10] is a widely used technique for producing hierarchical clusterings. Its popularity arises from the twin properties of speed and algorithmic simplicity. Algorithm 1 specifies the neighbor joining algorithm used in the evolution of data sets, UPGMA.

Algorithm 1: **Neighbor Joining**

Input: A collection of taxa, as points
Output: A tree with the taxa as leaves
Details:
Initialize a set of data from the taxa
  each with weight one
While (number of data>1)
    Join the closest pair of points
      Replacing them with their weighted average
    The weight of the new point is the sum of
      the weights of the points joined to make it.
End While
Report Tree

Amanda Saunders and Daniel Ashlock are with the Department of Mathematics and Statistics at the University of Guelph in Guelph, Ontario Canada, {asaunders|dashlock}@uoguelph.ca

Sheridan Houghten is at the Department of Computer Science at Brock University in St. Catherines, Ontario, Canada, shoughten@brocku.ca

There are many variations of neighbor joining, with a major point of variation being how to combine points. In [9] it was shown that three of the most popular methods, averaging, complete linkage, and centroid based, *all* exhibit substantial instability. The version shown in Algorithm 1 replaces points with their weighted average; other methods retain sets of points that have been joined and measure distance by closest (or furthest) pairs in the set. The averaging method is algorithmically fastest and so most widely used. This study will compare the stability of other methods of creating hierarchical clusters to the averaging method or neighbor joining. Examples of the kind of trees that might result are shown in Figure 1.

An evolutionary algorithm is used, in conjunction with a stability measure defined subsequently, to evolve data sets that are exceptionally stable and exceptionally unstable with respect to the averaging neighbor joining method. The stability of these data sets under a novel method called *bubble clustering* are then examined. Bubble clustering, defined in [9], has been shown to be more stable than neighbor joining on data sets not selected for a particular level of stability. Examining its behavior on exceptional data sets will provide additional evaluation and context for the value of the method.

A. The Stability Crisis

In [7], several pitfalls of clustering biological data are studied, and it is mentioned that ensemble clustering (which they define as clustering different ways, perturbing either the data or the parameters) often avoids these pitfalls. This notion of perturbing the data relates closely to our measure of stability.

In [3] a preliminary study demonstrated that neighbor joining of data sets created by placing 20 points uniformly at random in the unit square was violently unstable under the deletion of single data points. In [9] the presence of this instability was verified and also shown to exist in multiple types of neighbor-joining algorithm. The reason for this instability is that the absence of a single point can remove an aggregation that moves an already aggregated sub-tree of the clustering into a different part of the space; the missing point then has a cascade effect on subsequent neighbor joinings. Associator-based algorithms like bubble clustering employ ensembles of information derived from the position of multiple points, all of which are assessed before any points are moved.

The remainder of this study is structured as follows. Section II develops and justifies the tree-stability metric. Section III explains associator clustering and bubble clustering. Section IV describes the evolutionary algorithm used to generate

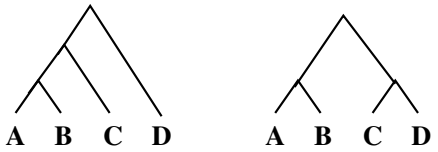


Fig. 1. A pair of trees on the same taxa with different connectivity.

the extreme data sets. Section V gives the experimental design, Section VI gives and discusses results, and finally Section VII draws conclusions and outlines possible next steps.

II. The Instability Measure

In biology a subtree of a taxonomy, itself displayed as a tree, is called a *clade*. For any two taxa used to build a tree (taxa are the leaves of the tree) there is a smallest sub-tree that contains them. We call this subtree the *minimal containing clade* (MCC) of the two taxa. If there are n taxa there are $\binom{n}{2}$ pairs of taxa and so $\binom{n}{2}$ minimal containing clades. The *size* of a clade is defined to be the number of leaves in its subtree or, alternatively, the number of taxa in the clade. In [3] it was shown that the map from a tree to its corresponding vector of sizes of minimal containing clades is an injection of the space of trees into Euclidean space. From this it was demonstrated that the standard (Euclidean) distance between minimal containing clade vectors places a metric structure on the space of trees. Example 1 demonstrates the calculation of the distance between two trees. We call this distance the *minimal containing clade*-distance (MCC-distance).

Example 1: Examine the two trees on the set of taxa $\{A, B, C, D\}$ shown in Figure 1. Since there are four taxa there are six pairs of taxa. For each pair the size of their minimal containing clade is the number of taxa in the smallest sub-tree that containing them. Tabulating these MCC sizes we obtain:

Taxa pair	Tree1 MCC Size	Tree 2 MCC Size
AB	2	2
AC	3	4
AD	4	4
BC	3	4
BD	4	4
CD	4	2

From this information we can derive the MCC-vectors for the two trees as (2,3,4,3,4,4) and (2,4,4,4,4,2). Treating these vectors as points in six-dimensional space, we obtain a distance between the trees of:

$$\sqrt{(2-2)^2+(3-4)^2+(4-4)^2+(3-4)^2+(4-4)^2+(4-2)^2}$$

or $d(tree_1, tree_2) = \sqrt{6}$.

The problem addressed in this study is the stability of trees to the addition or removal of single taxa. We adopt the instability measure from [3]. This measure uses the distance between two trees based on different methods of removing one taxa. If we have an existing tree, then we may simply remove a taxa and smooth over the point of removal *or* we may delete that taxa from the data set and re-run the tree-building algorithm. Both methods produce a new tree on the original taxa, with the exception of the removed taxa. These two trees are called the *removal tree* for taxa A, R_A, and the *rebuilding tree* for taxa A, B_A. If $\mathcal{T}$ is the set of all taxa appearing in the tree the stability measure is:

$$Instability = \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} d\left(R_x, B_x\right) \tag{1}$$

Where $d(\cdot, \cdot)$ is the MCC-distance between trees. In other words the instability measure is the average, over all taxa, of the MCC-distance between the removal and rebuilding trees for the taxa. This measure indicates complete stability with an instability value of zero, while larger numbers indicate higher levels of instability.

It is necessary to defend the MCC-distance as a good choice for measuring the instability of a hierarchical clustering algorithm. The MCC-distance is an excellent metric for measuring tree stability because it is based entirely on how close trees place each pair of taxa. This size of the minimal containing clade of a pair of taxa represents the hypothesis embedded in the tree about how closely related those taxa are. This means that the MCC-distance responds to, and responds only to, disagreements between trees about the relatedness of taxa.

III. Associator Clustering and Bubble Clustering

This section defines both the theoretical background and the specific type of clustering to which the stability of neighbor joining is compared. An *associator* is any measurement that establishes some kind of similarity between two objects. An example of an associator can be derived from k-means clustering [5]. If two points are placed together in the same cluster in a given instance of k-means clustering, then we can associate them more strongly. An associator can use a *quality measure* that establishes how much an associator strengthens the connections between two objects. One of the first associators implemented used k-means clustering with a reward of 1 for being in the same cluster, for example.

In order to perform hierarchical clustering, many associators are assessed, creating degrees of association between pairs of data points. The resulting matrix of pairwise associations for all data points, termed the *association matrix*, is populated by summing the outcomes of all the associators assessed. Once pairwise association strengths are available for the data set, joining most closely associated pairs permits the construction of a hierarchical clustering from the associator matrix in a manner described subsequently.

Bubble clustering uses associators in the form of spheres in the data space. The algorithm for bubble clustering is given as Algorithm 2. It places spheres in the data space and then,

for each pair of points in the sphere, increases the connection between them by the reciprocal of the number N of points in the sphere. In general a sphere that contains more points yields less information about the similarity of points within the sphere. The reward, i.e. increase in association strength, of $\frac{1}{N}$ means that togetherness in a sphere is rewarded adaptively and sensibly relative to the number of points in the sphere.

Algorithm 2: **Bubble Associator Matrix Generation**

Input: data points.
Output: An associator matrix.
Details:
Find the min, max distance between data points.
Repeat
  Generate R in the range $[min, max]$
  Pick a data point p
  Count number N of points within R of p.
  Mark all the points counted
  For each marked pair (a, b) of points
    Associate(a, b) with strength $1/N$
Until (Desired number of samples examined)

Once the associator matrix is constructed, a tree is constructed as follows. All taxa are initially placed in singleton sets. The two remaining sets which each contain one member of a pair with the highest remaining association strength are then joined, replacing the two sets with their union; high numbers associated points already within a single set are ignored. This algorithm is a neighbor joining algorithm. The critical difference is that the numbers used to perform the joins are all computed prior to any joining rather than on-the-fly during joining.

In the analysis of experiments three versions of bubble clustering are used, B_1, B_2, and B_3. These versions differ only in the number of bubbles used to create the associator matrix; B_n uses $D \cdot 10^n$ bubbles where D is the number of points in the data set.

IV. The Algorithm for Evolving Data Sets

The representation used for evolving data sets is an array of k points in n dimensions, realized as an $n \times k$ array. Points are placed within the unit hypercube $[0, 1]^n$ as appropriate to their dimension. The points are treated as atomic objects for crossover, employing two point crossover on the list of points. Mutation is controlled by the *maximum number of mutations* (MNM) parameter. For each data set produced by crossover, the algorithm selects from 1 to MNM mutations uniformly at random. The individual mutations consist of generating a new point, uniformly at random, in place of an existing point.

The algorithm is steady state [11] with selection and replacement performed by size seven single tournament selection. This model of evolution chooses seven population members and replaces the two worst in the group with copies of the two best. The copies are then subjected to crossover and mutation. The algorithm uses the instability measure on averaging-based hierarchical clustering as its fitness function; in some experiments instability is maximized, in others it is minimized.

V. Experimental Design

A parameter setting study was performed to study the effect of population size and mutation rate on the performance of the algorithm. A full factorial on population sizes 10, 32, 100, 320, and 1000 and the values 1, 3, and 5 for the MNM parameter was performed for both maximization and minimization of instability. The parameter study was performed on collections of 20 points in two dimensions. Four additional experiments were performed as well, one pair maximizing and minimizing stability on sets of 20 points in five dimensions and one pair maximizing and minimizing instability on 40 points in two dimensions. Each experiment consisted of 30 replicates of the evolutionary algorithm run for 10,000 instances of tournament selection updating the population. Table I gives the index number of the experiments.

Maximization			Minimization		
index	popsize	MNM	index	popsize	MNM
1	10	1	16	10	1
2	32	1	17	32	1
3	100	1	18	100	1
4	320	1	19	320	1
5	1000	1	20	1000	1
6	10	3	21	10	3
7	32	3	22	32	3
8	100	3	23	100	3
9	320	3	24	320	3
10	1000	3	25	1000	3
11	10	5	26	10	5
12	32	5	27	32	5
13	100	5	28	100	5
14	320	5	29	320	5
15	1000	5	30	1000	5
31	10	1	32	320	1
33	10	1	34	320	1

TABLE I
INDEX NUMBERS AND PARAMETERS FOR EXPERIMENTS FOR BOTH MINIMIZING AND MAXIMIZING INSTABILITY.

VI. Results and Discussion

The results for the parameter studies comprised by Experiments 1-15 and 16-30 are shown in Figure 2. For both maximization and minimization, the best performance comes from the lowest mutation rate, which suggests a relatively rugose fitness landscape in which crossover is an important search operator.

The best population size, unlike mutation, turned out to be different for maximization and minimization of instability. For maximization the best population size was 10 or 32 with relatively little difference between the two; large population sizes returned sharply worse performance. The parameter study minimizing instability had its best results at a population size of 320 with both extreme population sizes returning worse results. The results of the parameter studies were used to choose the parameters for experiments 31-34.

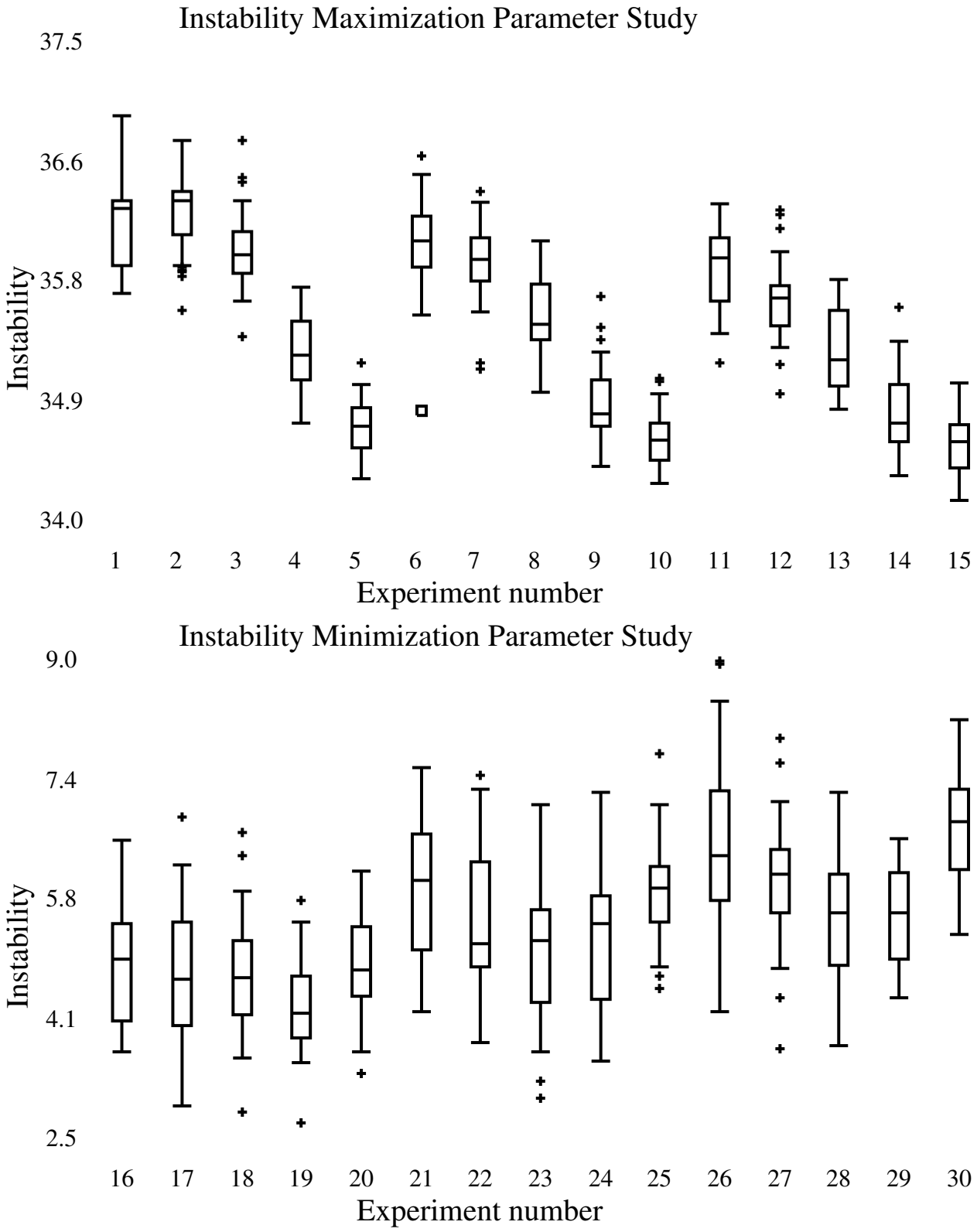


Fig. 2. The upper panel shows the distribution of maximized instabilities from best-of-run data sets for the 20-point, 2-dimensional, data set evolution experiment. The lower panel shows the distribution of minimized instabilities for the same type of data sets. Groups of five from left to right have the same mutation rate with higher mutation rates to the right while the groups of five are ordered from left to right by increasing population size.

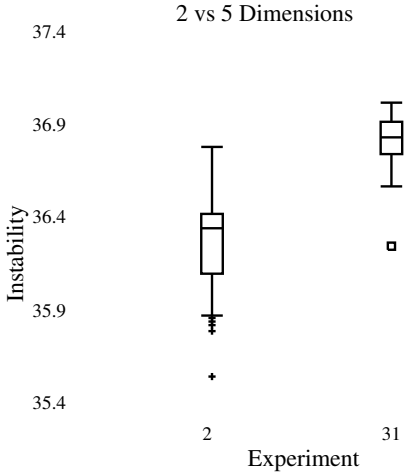


Fig. 3. A comparison between the best maximization result in the parameter study in two dimensions (Experiment 2) with the experiment using the same parameters in five dimensions (Experiment 31).

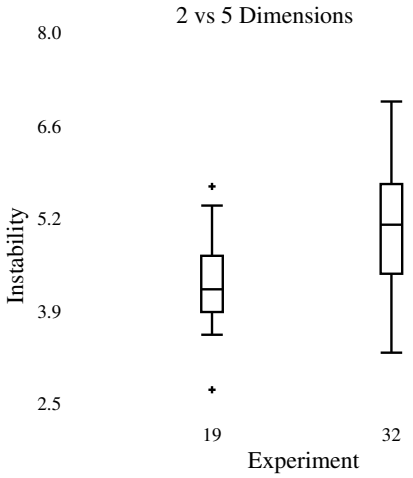


Fig. 4. A comparison between the best minimization result in the parameter study in two dimensions (Experiment 19) with the experiment using the same parameters in five dimensions (Experiment 32).

Figures 3 and 4 show the comparison, for maximizing and minimizing stability respectively, of changing from two to five dimensions. The problem of maximizing stability became considerably easier in five dimensions while the problem of minimizing instability became substantially *more* difficult in five dimensions. This result agrees with results in [9] that show the instability of trees produced with neighbor joining increases with data dimension.

Experiments 33 and 34 re-run the parameter values found in the parameter study on 40 points in two dimensions. Exemplary evolutionary runs for a maximization and minimization of instability appear in Figure 5. These plots echo the results from the parameter study as a whole; maximizing instability is much easier than minimizing it and the amount of change away from the initial population is greater when minimizing instability. The initial study [3] performed a sampling study that demonstrated that high levels of instability are the norm in randomly generated sets of data; this study supports this result. The distance from a random initial population to high instability is smaller than the distance to high stability.

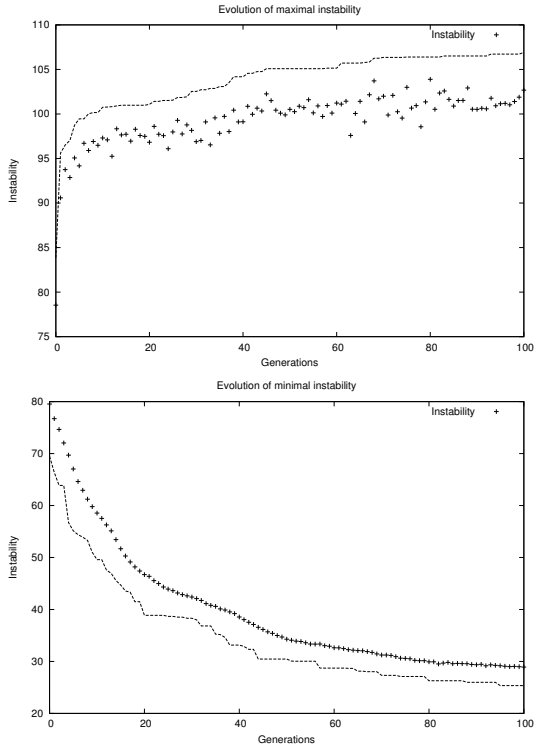
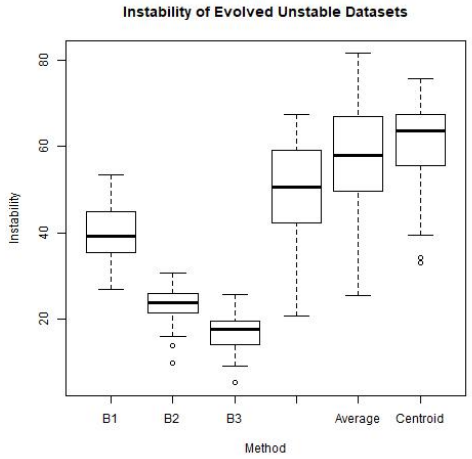


Fig. 5. The upper panel shows the single evolutionary run from Experiment 2 that achieved the best maximization result while the lower panel displays the best instability minimization result from Experiment 19. These experiments are the ones that yielded the best overall results in the parameter study.

The population sizes that are empirically best, 10 for maximization and 320 for minimization, suggest that minimizing instability is the harder problem. This suggests that instability is caused by simple structures involving a few points. If this is the case then low instability requires these structures be avoided by all small subsets of points. This model of the asymmetric difficulty of finding the two stability extremes is consistent with the observed greater difficulty of minimizing instability.

A. Relative Stability of evolved data with different algorithms

The most-fit data sets from experiments 2, 19, 31, 32, 33 and 34 were further processed to evaluate the relative instability when different clustering methods were applied. In all experiments the weakest bubble clustering coverage level fails to produce more stable trees but both B2 and B3 produce more stable trees than all tested versions of hclust(). (All p-values for comparisons between B2, B3 and the versions of hclust() had $p < 0.001$ except the difference between B2 and H1 with $p = 0.00794$ in Experiment 19). These three comparisons are shown in Figures 6-8. The three versions of bubble clustering described in Section III are compared with



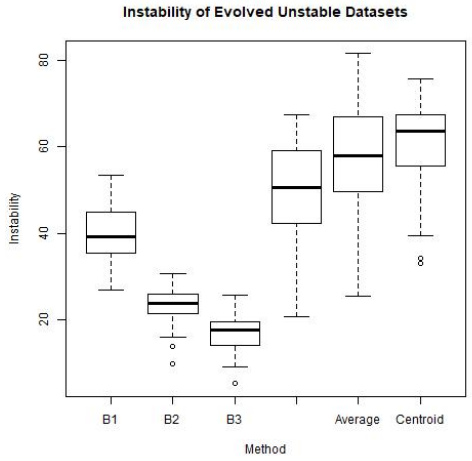
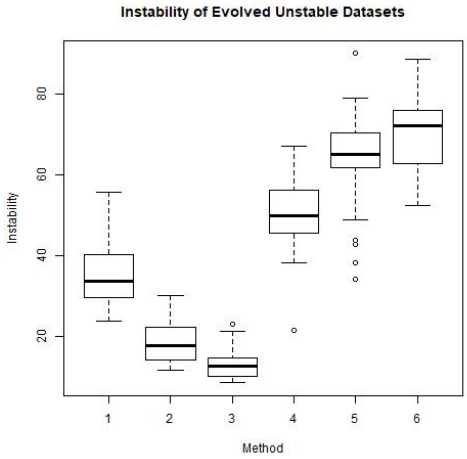


Fig. 6. Average Instability of Experiment 2 (upper) and 19 (lower) data-sets evolved to be respectively unstable and stable with UPGMA methods when various clustering methods were applied. These experiments have 20 points in two dimensions.



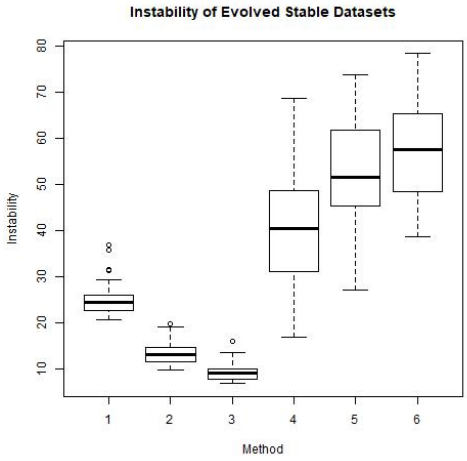


Fig. 7. Average Instability of Experiment 31 (upper) and 32 (lower) data-sets evolved to be respectively unstable and stable with UPGMA methods when various clustering methods were applied. These data sets have 20 points in five dimensions.

the R package hclust() using complete linkage, average and centroid methods for all six comparisons.

In agreement with the previous studies, increasing the level of bubble coverage produces more stable trees. It should be noted that the average instability of the evolved unstable data-sets when hclust methods are applied is greater than their evolved stable counter-parts. Various parameter differences can grow or shrink this difference. The results suggest that the patterns observed during the initial testing of the instability behavior of hclust() and bubble clustering are holding true even in data sets evolved to exhibit extreme stability or instability. As in the earlier study, increasing the number of dimensions of the data increases the instability generated using hclust() methods while improving the stability of bubble clustering results.

VII. Conclusions and Next Steps

This study replicates, extends, and confirms an earlier study [3] demonstrating that a neighbor joining algorithm that uses averaging of the mean positions of data during clustering is highly unstable. The results of the current study substantially extend the number of parameters for which the algorithm for evolving extremal data sets is tested and also confirm that standard neighbor joining and bubble clustering respond in opposite directions to increasing data dimension as found in [9].

The working hypothesis to explain the instability of popular neighbor joining algorithms is that moving the position of the objects being clustered on the fly during clustering is a source of instability and therefore a bad idea. This hypothesis is supported implicitly by the current study simply because bubble clustering first computes the data used to join groups of points and then performs the neighbor joining on the pre-computed object.

It is worth noting that *any* associator clustering will share with bubble clustering this property of first computing the information needed to perform the clustering and then performing the clustering on effectively immobile data points. The stability, while experimentally demonstrated for bubble clustering, is probably an intrinsic property of any associator

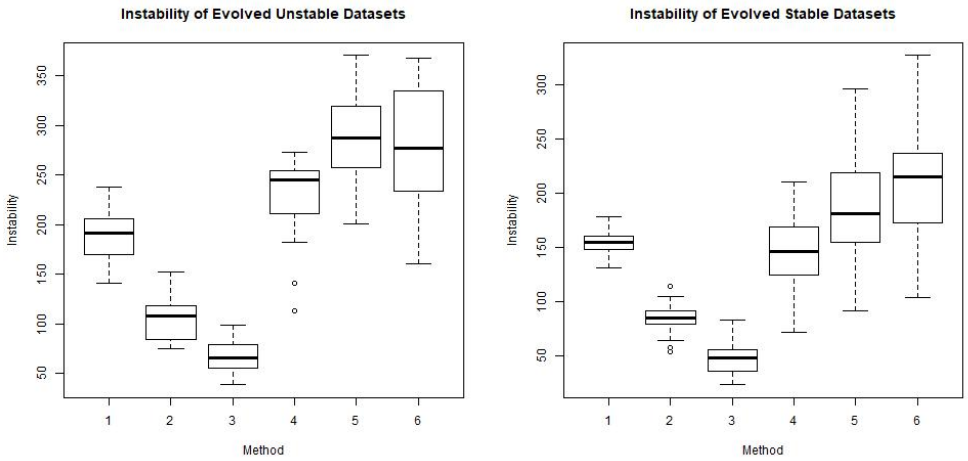


Fig. 8. Average Instability of Experiment 33 (left) and 34 (right) data-sets evolved to be respectively unstable and stable with UPGMA methods when various clustering methods were applied. These data sets have 40 points in two dimensions.

clustering. For example, the work in [4], that used a concept of associator matrices that were then used as adjacency matrices to build trees, also demonstrated significantly improved stability.

Another nice feature of bubble clustering that is intrinsic in most associator clustering methods is the ability to spend effort to improve quality. The more bubbles are used, the greater the stability. In [9], bubble clustering was run at 10,000 bubbles per point and it was shown that there is a diminishing return in increased stability: 1,000 bubbles per point and 10,000 exhibited similar levels of stability. The only way an associator method would not have this property would be if the number of possible associators were quite small.

The speed of standard neighbor joining – that it requires only $n-1$ fundamental operations (joins) to assemble a tree on n points – is also its Achilles heel. The use of additional associators permits bubble clustering to approximate more and more closely a fixed, underlying object that is minimally disrupted by the removal or addition of a few taxa. In contradistinction, the contingent nature of joins that can be deflected by the addition or removal of taxa is the source of instability in neighbor joining.

A. The Case for Stability as a Quality Measure

The stability measure used in the study defines stability in terms of the degree to which a hierarchical clustering algorithm changes its mind about the relatedness of all pairs of taxa, except for pairs involving a removed taxa, when that taxa is removed. The measure is computed over the entire data set by averaging over all taxa, removing each in turn. The size of a minimal containing clade is a good one-parameter estimate of the relatedness of two taxa, with smaller numbers suggesting higher relatedness.

If we assume that there is a true relatedness between pairs of taxa – something that would be true if the data were drawn from a clade of animals arising via natural evolution – then any good hierarchical clustering algorithm should return a close approximation to the true evolutionary relationship, unless the features chosen for similarity classification are inappropriate.

This criterion is a stinging indictment of any clustering algorithm that can substantially rearrange its opinion about relatedness when a single taxon is removed. The variations of hclust() used in this study and in [9] all exhibit high levels of instability and that instability grows with the dimension of the data. This means that adding additional information – measured characters describing the taxa – *degrades* algorithm performance for reasons that this study suggests are intrinsic to the geometry of Euclidean space.

Bubble clustering is slower than the various hclust() methods, but it also has a clear speed-quality trade-off. Additionally, the algorithm that transforms the associator matrix into the hierarchical clustering is similar – in both details and speed – to that used in standard hierarchical clustering. One can simply accumulate more associators when computing power is available and update the associator matrix incrementally.

B. Taxonomy and Appropriate Clustering Algorithms

A potential use of the instability measure – that is generic across clustering algorithms – is to check the instability of different types of data for different algorithms. If we think of clustering algorithms as agents and data sets to be clustered then agent-case embeddings [2] can be used to create dual family trees of algorithms and data types that permit the selection of the most appropriate algorithm to hierarchically cluster a given sort of data.

Agent-case embeddings require a score for each agent on each problem case. For this effort the score would take the form of a stability measure. The rows and columns of the matrix of scores, indexed by agents and data sets, form an injection of the clustering algorithms into Euclidean space where they can be bubble clustered or otherwise classified.

The columns form a similar injection for the data sets. The matrix, as a whole, is a collection of behavioral features of algorithms on sets of data that is potentially useful for a variety of analyses.

While there are good reasons to think that associator clustering algorithms are slower and higher quality than current popular neighbor joining algorithms, only bubble clustering and K-means multi-clustering [5] have been tested on data thus far. These two sorts of associator clusterings (K-means multi-clustering pre-dates the associator terminology) are the only ones evaluated among a substantial variety of possible associator-based clustering algorithms. The proposed classification scheme could be used to select among such algorithms.

C. Variations on Bubble Clustering

Bubble clustering chooses its bubbles by first estimating a reasonable range of radii for bubbles and then generating bubbles by choosing both a uniformly distributed radius within the computed range and selecting a data point at random to serve as the center of the bubble. Preliminary work suggests that revisiting the way the radius is chosen is potentially beneficial.

In the work on K-means multi-clustering [5], a large number of clusters were used. This meant that the diameter of the sets of points used for associating points was small – corresponding to a much smaller collection of potential radii for bubbles. One of the interesting features of K-means multi-clustering was that it could discover non-convex clusters in the data.

Initial experimentation with reducing the radii of bubbles used in bubble clustering demonstrates that the initial radius selection range – approximately uniformly at random within the diameter of the data – does not reproduce this property of detecting non-convex clusters but that reduction of the maximum radius chosen does.

Another obvious variation is to change the reward structure. At present, if N points are in a bubble, association is strengthened by $1/N$. Giving a more nuanced reward – e.g. more reward for being near the center of the bubble, would yield a different clustering algorithm, grist for the taxonomic mill of finding stable clustering algorithms for particular types of data.

D. The Point-packing Associator

A proposed point-packing associator leverages a type of clustering introduced in [1] based on point packings in a set. A web repository [1] gives the current best known values for the point packing problem into the unit square. For the original point packing problem the goal is to maximize the number of points packed into the unit square, subject to a minimum distance constraint that applies to all pairs of points. The work in [1] generalizes the set to more than the unit square in an application-driven fashion. To make an associator, the point packing is used to partition the data and then the elements of the partition each serve, individually, as associators.

In order to create an associator from a point packing we use the following scheme.

1) Points are packed into the space containing the data. Multiple packings are performed.
2) Given a point packing, data points are assigned to clusters based on which point in the packing they are closest to.
3) Using the resulting cluster partition of the data, the association of any two data points is increased by the reciprocal of the size of the cluster they belong to.

The association strength (quality measure) used for point packings is very similar to the one used for bubble clustering - the reciprocal of the number of associated members. The difference is that the bubbles are replaced by tiles of the Voronoi diagram [12] induced by the point packing. Using single bubbles permits weak, long-distance associations to form while use of a well-spaced point packing to generate Voronoi tiles will force associations to all be relatively short-range. The associations made by bubble clustering are independently chosen, while the point-packing associations between points come in blocks of mutually exclusive associations. It is not clear *a-priori* which technique will yield better results (higher tree stability), so this study compares the stability of both associator techniques and also compares them with averaging-based neighbor joining.

[1] **http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html**

REFERENCES

[1] D. Ashlock and S. Graether. Conway crossover to create hyperdimensional point packings, with applications. accepted to WCCI 2016, 2016.

[2] D. Ashlock and C. Lee. Agent-case embeddings for the analysis of evolved systems. *IEEE Transaction on Evolutionary Computation*, 17(2):227–240, 2013.

[3] D. Ashlock, T. vonKonigslow, and J. Schonfeld. Breaking a hierarchical clustering algorithm with an evolutionary algorithm. In C. Dogli *et. at.*, editor, *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 19, pages 197–204. ASME Press, 2009.

[4] S. Izekulova. Associator trees. Undergraduate thesis, Brock University, 2011.

[5] E.Y. Kim, S.Y. Kim, D. Ashlock, and D. Nam. Multi-k: accurate classification of microarray subtypes using ensemble k-means clustering. *BMC Bioinformatics*, 10(260):1–12, 2009.

[6] P. Legendre and L. Legendre. *Numerical Ecology, 2nd Edition*. Elsevier Science, Amsterdam, 1998.

[7] T. Ronan, Z. Qi, and K.M. Naegle. Avoiding common pitfalls when clustering biological data. *Sci. Signal.*, 9(432):re6–re6, 2016.

[8] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[9] A. Saunders. In search of more stable hierarchical trees: devising new algorithms to improve upon the stability of neighbor joining. Master's thesis, University of Guelph, 2017.

[10] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy; the Principles and Practice of Numerical Classification.* W.H. Freeman, San Francisco, 1973.

[11] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[12] G. Voronoi. Nouvelles applications des paramtres continus la thorie des formes quadratiques. *Journal fr die Reine und Angewandte Mathematik*, 133:97178, 1908.