Interactive Visual Clustering

Marie desJardins
James MacGlashan
MAPLE Laboratory
Department of CS&EE
1000 Hilltop Circle
Baltimore, MD 21250 USA
+1-410-455-3967
{mariedj, jmac1}@umbc.edu

Julia Ferraioli
Bryn Mawr College
101 North Merion Ave.
Bryn Mawr, PA 19010
+1-610-526-5358
jferraio@brynmawr.edu

ABSTRACT
Interactive Visual Clustering (IVC) is a novel method that allows a user to explore relational data sets interactively, in order to produce a clustering that satisfies their objectives. IVC combines spring-embedded graph layout techniques with user interaction and constrained clustering. This paper describes the IVC method, and gives experimental results on several synthetic and real-world data sets, showing that IVC yields better clustering performance than several alternative methods.

ACM Classification: I2.6 [Artificial Intelligence]: Learning. H5.2 [Information interfaces and presentation]: Graphical user interfaces.

General terms: Algorithms, Experimentation.

Keywords: Clustering, constraints, interaction, machine learning.

MOTIVATION
The goal of this research is to develop interactive clustering methods, which allow a user to partition a data set into clusters that are appropriate for their tasks and interests. The goal of traditional automated clustering is to partition a data set into clusters that have high intra-cluster similarity and low inter-cluster similarity. In general, there will be a single best clustering (or possibly several local maxima), which depend on the similarity metric used for clustering, the particular objective function being optimized by the clustering algorithm, and the search method.

In practice, however, the "best" clusters may also depend on the user's goals and interests. For example, when performing clustering in a collection of student data, an admissions officer may be looking for patterns in student performance, whereas a registrar might want to track enrollment patterns for different course offerings. The appropriate clusters will not be the same for these two users. An automated clustering method might find one of these clusterings, but not both.

Recent work on constrained clustering addresses the issue of discovering multiple target clusterings, using additional knowledge provided by a user. Constrained clustering is based on the insight that although users may not be able to explicitly state the criteria for their desired clustering, they can often provide partial knowledge about the nature of the clusters. This additional information is typically given in the form of pairwise constraints on cluster membership, which are used to guide the system towards the desired solution.

Ideally, the user would provide a few initial constraints to "seed" the clusters, then add constraints as necessary to adjust and improve the resulting clusters. The difficulty with doing this in practice is that the clusters are not always easy to understand, particularly in high-dimensional domains, where the "shapes" of the clusters are also high-dimensional and therefore difficult to visualize clearly.

In some domains, there may be other relational information in addition to the pairwise constraints. This relational information, which can be represented as edges in the data graph, provides additional similarity information. However, these relations are generally weaker than pairwise constraints: they do not strictly imply shared cluster membership, although they may indicate a higher cluster correlation between the connected instances. Most clustering algorithms take into account either the attribute information on the data instances, or the relational information between instances, but not both.[1]

Our goal is to allow a user to explore a large relational data set interactively, in order to produce a clustering that satisfies their objectives. We achieve this goal by combining spring-embedded graph layout techniques with user interaction and constrained clustering.

Specifically, we present a novel approach called Interactive Visual Clustering (IVC). In this approach, the relational data is initially displayed using a spring-embedded graph lay-

[1]Note that there has been some recent work on relational clustering [3, 24, 14, 19], including our own ongoing research in this area [1]. However, to our knowledge, none of the existing work explicitly combines relational data with pairwise constraints.

out. The user can then move groups of instances to different places on the screen in order to form initial clusters. A constrained clustering algorithm is applied to generate clusters that combine the attribute information with the constraints implied by the instances that have been moved. These clusters are then used to generate additional graph edges, which are combined with the relational edges to produce a new layout, in which instances are relocated closer to the clusters to which they appear to belong. Based on the new layout, the user can identify instances that are "misplaced" and move these instances into the correct clusters.

We show experimentally, using several synthetic and real-world data sets, that using IVC, we can converge to the target clustering significantly more quickly than either manual adjustment, spring-embedded layout alone, or clustering alone.

BACKGROUND

Our work combines and extends standard force-directed graph layouts and the PCK-Means constrained clustering method introduced by Basu and Mooney [2].

Force-Directed Graph Layout

Force-directed layout methods are among the most popular graph layout techniques [10]. We use a type of force-directed layout called *spring embedding* [6], as implemented in the Prefuse graph visualization system [17].

In spring embedding, nodes in a graph act on each other with two kinds of simulated forces, modeled on physical processes. The first force is a *node repulsion force* emitted from each node, simulating an inverse gravitational force. The repulsion force from a node exerts a "push" on every other node in the graph, with a magnitude inversely proportional to the square of the distance between the nodes. The second force is an attractive *spring force* that "pulls" (or pushes, if the nodes become too close) along the edges between the nodes. Each edge is modeled as a spring with an ideal *spring length*, and a *spring constant* (strength). The edge exerts a force on the nodes at either end, with magnitude proportional to the spring constant, and inversely proportional to the difference between the ideal length and the current length. If the edge is longer than the ideal length, the force is an attracting force; if shorter than ideal, the force is a repulsing force.

The spring-embedded layout is determined iteratively, by computing and summing all of the forces on each node, then moving the nodes incrementally in the direction of the net resulting force. This "settling" process is repeated until the layout reaches an equilibrium. In the resulting layout, nodes with edges between them tend to be situated near each other, whereas nodes without edges between them tend to be spread apart.

Constrained Clustering

Many different techniques for clustering exist, based on different models of the nature of the clusters to be discovered [8]. Perhaps the most commonly used clustering technique is the K-means algorithm [13], which iteratively assigns points to the nearest cluster, then recomputes cluster centroids, until a stable clustering is reached.

Recently, researchers in the machine learning community have developed methods for *constrained clustering*, in which users are able to provide additional information about the nature of the clusters. This information typically consists of pairwise constraints on individual data points, indicating that those points should be in the same cluster (*must-link* constraints), or should be in different clusters (*cannot-link* constraints). The earliest constrained clustering work, by Wagstaff et al., extended K-means by modifying the iterative assignment step to select the nearest cluster that satisfies the constraints.

Basu and Mooney [2] extended Wagstaff's work to use a generalized weighted penalty function. In their approach, PCK-Means, if a cluster assignment determined by the K-means method violates the assigned must-link or cannot-link constraints, then a penalty is assigned to the overall solution. This penalty is incorporated into the K-means objective function, so PCK-Means effectively searches for the cluster assignment that maximizes cluster coherence while minimizing the penalty for constraint violations. To perform constrained clustering in IVC, we are using an implementation of PCK-Means provided to us by the original authors as an extension to the Weka learning toolkit [21].

APPROACH

Our visual clustering paradigm consists of the following steps:

1. **Initializing the display**. We currently use Prefuse's spring-embedded graph layout algorithm to produce the initial display.
2. **Interpreting user actions.** As the user moves instances, pairwise cluster membership constraints are generated.
3. **Constrained clustering.** After each instance is moved, the new constraints are added to the constraint set, and a constrained clustering algorithm (PCK-Means) is used to produce a new clustering of the data.
4. **Updating the display.** Using the clusters produced, the display is updated so that the new clusters are visually apparent.

The following subsections describe our approach to Steps 2 and 4. We then discuss mechanisms for simulating user behavior for our experiments.

Interpreting User Actions

When the user moves an instance, it is "pinned" in place, and is not affected by the spring-embedded layout. These pinned instances do, however, exert forces on the other instances in the graph.

The constrained clustering process begins after the user has moved two instances. To generate the constraints, the screen distance between each pair of moved instances is computed. If the instances are at least δ units apart (where δ is a user-adjustable parameter), they are considered to be in different clusters, and a *cannot-link* constraint is added between them. If they are less than ϵ units apart (where $\epsilon \leq \delta$ is another user-adjustable parameter), then they are considered to be in the same cluster, and a *must-link* constraint is added. If the

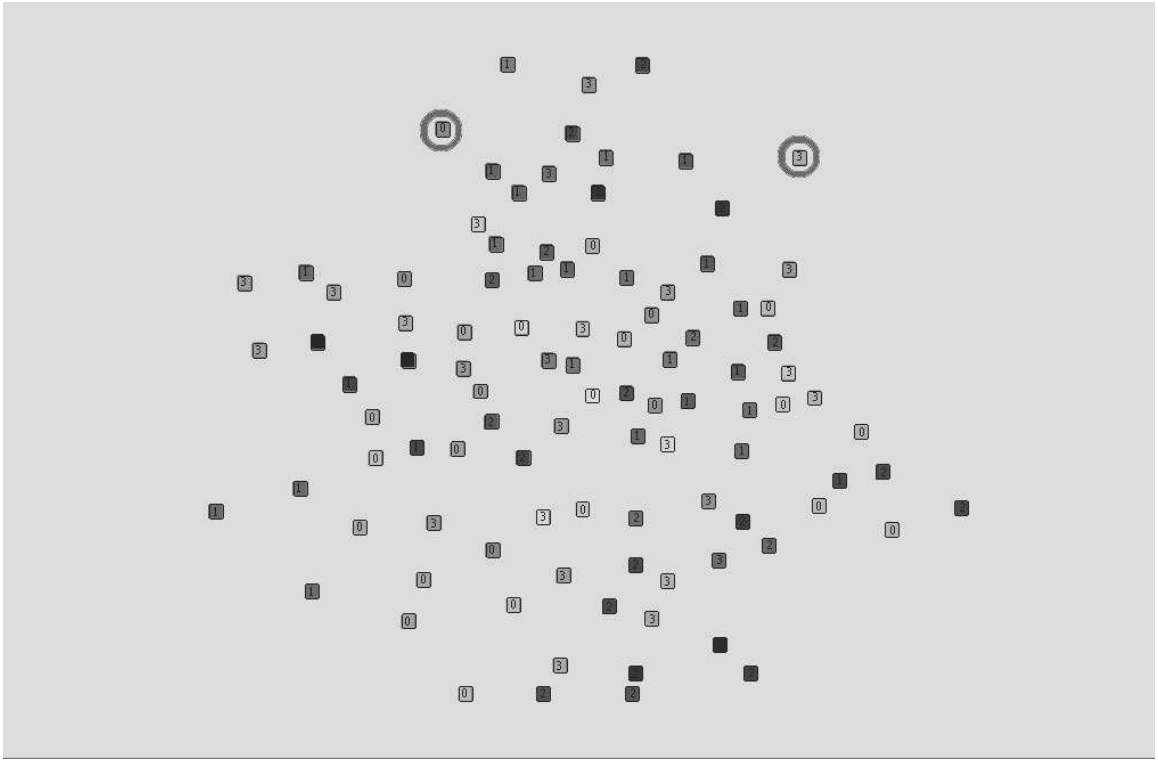

Figure 1: Initial display of the Overlapping Circles data set. The circled instances are the first two instances that will be moved by the user.

screen distance is greater than ϵ but less than δ, then the situation is ambiguous, and no constraints are generated.[2]

PCK-Means is then run on the data, using all of the constraints generated thus far, and a new clustering is produced. Note that the distance metric for PCK-Means is Euclidean distance in the attribute space, *not* the screen distance used to generate constraints.

Updating the Display

Once the new constraints have been generated, and a new clustering produced, the display must be updated to reflect the groupings inherent in the new clustering. Ideally, the relational structure of the data should also be preserved. If the relational structure is correlated with cluster membership, then the graph edges and the cluster membership edges will reinforce each other, leading to rapid convergence of the interaction to the correct clusters.

To update the graph, we adapt an approach described by Brockenauer and Cornelson [4] for visualizing clusters in graphs. First, a new "dummy" node is generated to represent the center of each cluster. This node is located at the center of the instances that were moved into that cluster (i.e., the transitive closure of a set of instances with must-link constraints between them). Next, a *cluster edge* is added between this cluster center and every instance assigned to that cluster. The relational edges use Prefuse's default spring constant ($2.0x10^{-5}$). Cluster edges are set to have a spring constant equal to twice the default ($4.0x10^{-5}$). As a result, the cluster edges have a more significant effect on the layout than the relational edges, but do not completely dominate the layout.

The spring-embedded layout is then invoked on the combined graph (i.e., the graph with both the relational edges from the original data set and the new cluster edges). The resulting graph is displayed, but only the relational edges are shown to the user, and the cluster center nodes are not drawn. (It would be possible to also show the cluster edges, but this makes the graph very cluttered and obscures the relational edges.)

Simulating the User

We have not yet run formal experiments on human users. For the experiments reported here, we simulate user behavior using one of two heuristics for instance selection: *random* and *farthest-first*. The random instance method simply selects a random instance to move at each step. The farthest-first method selects the instance that is farthest (on the screen) from its correct cluster. The intuition behind the latter heuristic is that the user will be most likely to notice anomalous instances — that is, the instances that appear farthest from where they should be. For both node heuristics, we use predefined locations (near the screen corners) for the cluster centers. Instances are moved to this location, with a small random (x, y) offset.

In the experiments with force-directed layout but no clustering, after each instance is moved, the layout is allowed to "settle" to an equilibrium before the next instance is moved.

[2] In our experiments, ϵ is set to 227 pixels, and δ is set to 450 pixels.

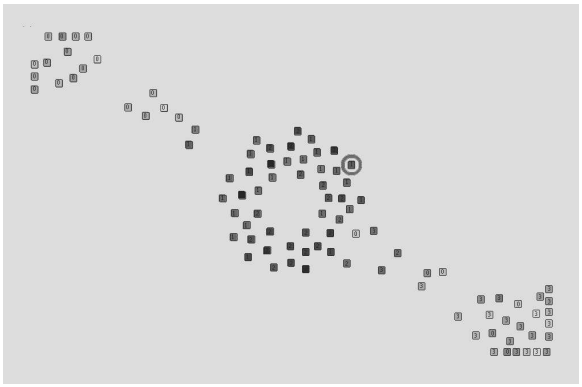

Figure 2: Layout of the Overlapping Circles data set, after the two instances shown in Figure 1 have been moved. The circled instance will be moved next.

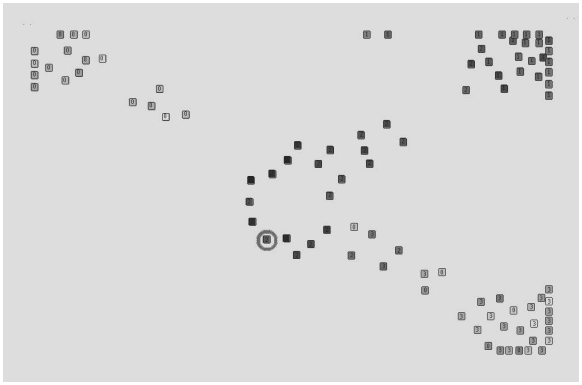

Figure 3: Layout of the Overlapping Circles data set after three instances have been moved. The circled instance will be moved next.

System Operation

A series of screenshots is shown in Figures 1 to 5. In this sequence, a user is moving nodes to their target clusters. Figure 1 shows the initial display of the synthetic Overlapping Circles data set. The Overlapping Circles data set is described in "Data Sets," below. For purposes of illustrating the process, the colors of the nodes and numeric labels indicate the "true" (target) cluster membership. In order to simplify the displays for these small-scale screen captures, the relational edges are not shown. Notice that nodes from all of the clusters are interspersed in the display.

The circled nodes in Figure 1 are the first two nodes chosen by the user. The resulting display is shown in Figure 2. Here, the upper left and lower right clusters (where the first two nodes were placed) are starting to become apparent. Figures 3 and 4 show the display after the third and fourth nodes are moved. In Figure 4, all four clusters can be seen; however, there are still a number of "ambiguous" nodes in the center of the display, which are not clearly associated with any one cluster.

Figure 5 shows the display after 14 nodes have been moved. At this point, as seen in the results in Figure 9, the Adjusted Rand Index using IVC is around 0.9, meaning that most of the instances are grouped correctly into their target clusters.

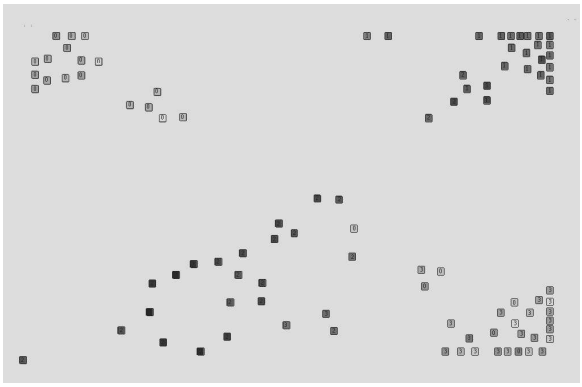

Figure 4: Screen display of the Overlapping Circles data set after four instances have been moved.

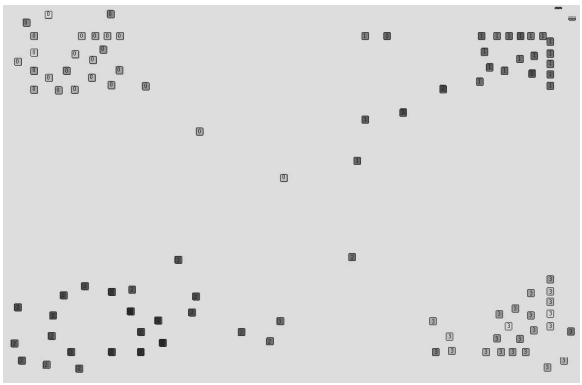

Figure 5: Screen display of the Overlapping Circles data set after 14 instances have been moved.

Visually, the clusters are very distinct, with only a few nodes scattered between the clusters.

METHODOLOGY

We compared our Interactive Visual Clustering method to several alternative approaches. The five approaches we tested are shown in Table 1. "Layout?" indicates whether force-directed layout is used. (If not, the layout only changes when instances are explicitly moved by the user.) "Clustering?" indicates whether constrained clustering is used. If so, each time an instance is moved, a new clustering is computed, and cluster edges are updated. If not, no cluster edges are used in the layout. "Heuristic" indicates the instance movement heuristic: either Farthest-First or Random.

Note that the fourth approach (Clustering Baseline) is equivalent to simply doing standard constrained clustering with random constraints, since the layout position is not taken into account in selecting which instance to move.

We hypothesize that the farthest-first instance heuristic will improve performance faster than moving random instances, since each instance moved should provide the largest possible improvement in the clustering.

Clustering is expected to perform faster than without clustering because it results in an explicit model of the user's target clustering, incorporating the feedback provided by the must-

Approach	Layout?	Clustering?	Heuristic
Manual Baseline	No	No	Random
Layout Baseline	Yes	No	Random
Layout + FF	Yes	No	Farthest-First
Clustering Baseline	Yes	Yes	Random
Interactive Visual Clustering	Yes	Yes	Farthest-First

Table 1: The five approaches that we tested empirically.

link and cannot-link constraints.

Force-directed layout should perform better than manual layout because of the relational edges in the data set. These edges already tend to cluster the instances visually, because they pull together instances that are related to each other. Therefore, when an incorrect instance is moved to its target cluster, it should also pull similar instances towards that cluster, resulting in the possibility of multiple instances being moved to the correct group. Furthermore, the cluster edges exert an even stronger influence (because their spring constant is higher), so as the clustering algorithm receives more constraints, the layout increasingly reflects the learned clustering. Therefore, the farthest-first heuristic primarily utilizes the relational knowledge from the data set when there are few constraints (few instances moved), but primarily utilizes the clustering structure when there are many constraints (many constraints). As a result, the interaction shifts from creating an initial clustering towards repairing the learned clustering over time.

Note that the IVC paradigm relies on an assumption that the relational edges are correlated with the cluster membership of the instances. If there is no such correlation, then these edges may not be helpful, or could even hinder performance.

To measure the performance of the alternative approaches, we use the Adjusted Rand Index (ARI) [7]. The ARI is used to evaluate how close a given clustering is to the "correct" or target clustering. The Rand Index [18] measures the proportion of clustering matches. (A "match" is a pair of instances that are either grouped together in both the learned and the target clustering, or grouped separately in both the learned and the target clustering.) Using C_i to indicate the cluster labeling of instance i in the target clustering and $C\prime_i$ to indicate i's cluster labeling in the learned clustering, we indicate the number of same-cluster matches as

$$M_{\textit{same}} = |\{i, j : ((C_i = C_j) \wedge (C\prime_i = C\prime_j))\}|$$

and the number of different-cluster matches as

$$M_{\textit{diff}} = |\{i, j : (C_i \neq C_j) \wedge (C\prime_i \neq C\prime_j))\}|.$$

Then the Rand Index is given by:

$$\begin{aligned} RI &= \frac{\text{\# matches}}{\text{\# pairs}} \\ &= \frac{M_{\textit{same}} + M_{\textit{diff}}}{\binom{n}{2}} \end{aligned}$$

The Rand Index penalizes partitions with more clusters, so the Adjusted Rand Index is often used instead. The ARI normalizes the Rand Index to adjust for the number of clusters, by comparing the expected number of matches to the observed number of matches [7]. The ARI is bounded between 0 and 1. An ARI of 1 means that all instances are correctly clustered. We use the ARI implementation provided with the Weka system [21].

In the experimental results, clustering performance is always shown as a function of the number of instances moved. Both the layout and the cluster assignments use a random initialization step, so for each experiment, we show the average performance over 20 runs.

Data Sets

We used five data sets to test our hypotheses: two synthetic data sets (Circles and Overlapping Circles), the Iris data set from the UC Irvine Machine Learning Repository [15], and two different data sets involving amino acid information, referred to as Amino Acid Indices and Amino Acid. The amino acid data sets include relational edges already, as explained later. For the synthetic and Iris data sets, we tested three methods for edge generation, resulting in three versions of each data set: one with no edges, one with edges generated by nearest-neighbor comparisons, and another version with edges generated probabilistically.

Nearest-neighbor edge generation creates an edge between each instance and that instance's nearest neighbor (using Euclidean distance in the attribute space). For data sets whose cluster membership is strongly related to the instances' distribution in Euclidean attribute space, this will result in edges that are well correlated with cluster membership. Nearest-neighbor edge generation results in a number of edges equal to or less than the number of instances (since one edge is created for each instance, but some pairs of instances may be each others' nearest neighbor, so only a single edge is added for the pair).

Probabilistical edge generation uses knowledge of shared membership in the true clusters to generate the edges. Specifically, for each pair of instances, if the instances belong to the same cluster, then an edge is created between them with probability 0.2. If the instances do not belong to the same group, than we an edge is created between them with probability 0.05. This process results in a denser graph than nearest-neighbor edge generation: the expected number of edges is $O(\frac{N^2}{k})$, where N is the number of nodes and k is the number of clusters. The actual expected number of edges depends on the distribution of nodes among the clusters. For k equal-sized clusters, the expected number of edges is ap-

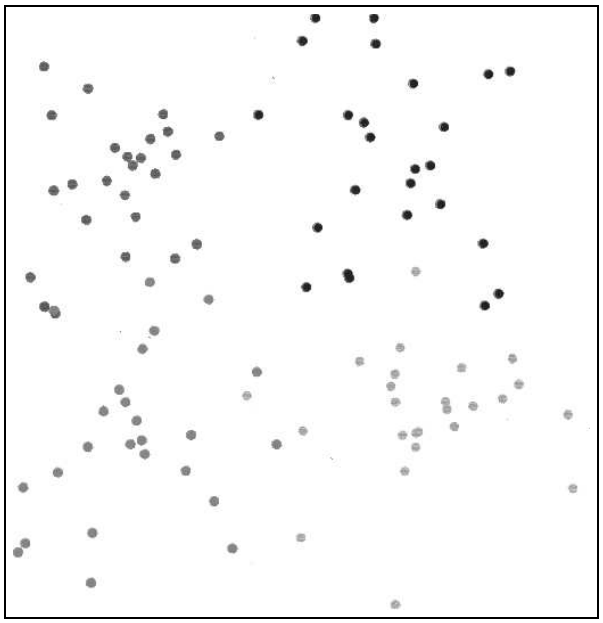

Figure 6: 2D view of the Overlapping Circles data set.

proximately $\frac{N^2}{8k}$.

Circles The synthetic data sets are simple low-dimensional clusters that are included as benchmarks for the different approaches.

The synthetic Circles data set includes 120 instances in two distinct clusters. These clusters are generated by positioning circles of radius 50 at [50,50] and [150,150] on the (x, y) plane. Fifty points are randomly selected from inside each circle, and assigned to the corresponding cluster. Twenty additional "outlier" instances are generated by randomly sampling betweeen the bounding circles. THese outliers are then assigned to the nearest cluster. The two attributes for each instance are just the (x, y) positions.

Because these clusters do not overlap and are well separated, in the nearest-neighbor version of this data set, there are no edges between instances from different cluster origins. In other words, if two points have an edge between them, they are in the same cluster. In the probabilistic-edge version of the data, there are more edges, some of which join instances of different clusters.

Overlapping Circles The Overlapping Circles data set includes 100 instances in four overlapping clusters. This data is generated by creating random points from a uniform distribution within the radius of four circles—corresponding to the four clusters—whose centers lie on another circle's radius at each 45-degree mark. As shown in Figure 6, the four clusters overlap each other. Therefore, some of the instances' nearest neighbors in Euclidean space can be from a different cluster. In this data set, both the nearest-neighbor and the probabilistic edges sometimes connect instances from different clusters.

Iris The Iris data set is a widely used classification database from the UC Irvine Machine Learning Repository [15] The original data set consists of 150 instances; we selected 99 of these instances to create our data set by choosing 33 instances randomly from each cluster. Each instance is described by four numeric attributes (sepal length and width, petal length and width). The three clusters correspond to the three classes provided with the original data set (three different species of irises: Iris Setosa, Iris Versicolour, and Iris Virginica). This data set is known to be a difficult one for most clustering algorithms, because two of the classes are linearly separable from each other, but the third is not.

Amino Acid Indices The amino acid data set is a subset of the AAIndex database [11, 9]. Our data set is based on Version 6.0 of the AAIndex database, which includes 494 *indices*, each of which measures a chemical property of amino acids. Each instance in the AAIndex database has twenty attributes, corresponding to the values of this index for each of the twenty amino acids used in the standard genetic code. Tomii and Kanehisa [20] identified six clusters of indices in the original database: A (measures of alpha and turn propensities), B (beta propensities), C (composition), H (hydrophobicity), P (physiochemical properties), and O (other). We use 100 of these indices, selected randomly from the A and H classes; the A/H classification is also used as the target clustering.

The edges in this data set were determined by measuring the correlations between the instances, then reducing the edges to a minimum spanning tree.

Amino Acid In this data set, the attributes and instances are inverted from the Amino Acid Index data set. The Amino Acid data set includes twenty instances—one for each amino acid—whose attributes are the amino acid's chemical properties. Twenty-five of the 100 indices from the Amino Acid data set are used as attributes. We first removed binary indices, which do not yield good clustering performance. Since many of the indices are minor variations of the same basic measurement, we then asked a domain expert to select 25 indices that measured relatively "orthogonal" (uncorrelated) properties.

Edges were added to the data set based on three properties of amino acids: acidic side chains, basic side chains, and cyclic hydrocarbons. Edges are placed between pairs of instances that share one or more of these properties.

The target clustering has three clusters, also manually identified by our domain expert: polar, non-polar, and both. Polar amino acids show asymmetrical electron charge on the amino acid side chain. Non-polar amino acids show symmetrical electron charge. Amino acids that have long side chains with both polar and non-polar regions are grouped into the "both" cluster.

RESULTS AND DISCUSSION

Overall, our experimental results support our claim—that Interactive Visual Clustering provides improved clustering performance, compared to the alternative approaches we tested. However, the Amino Acid Index data set does not yield the expected results, highlighting some of the open challenges.

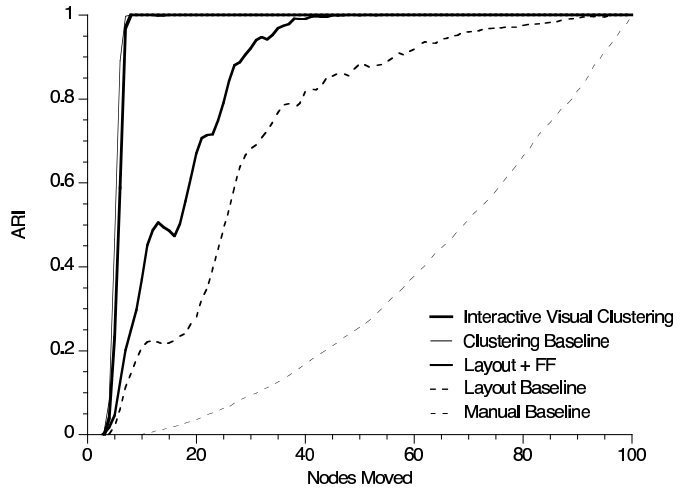

Figure 7: Experimental results on the Circles data set.

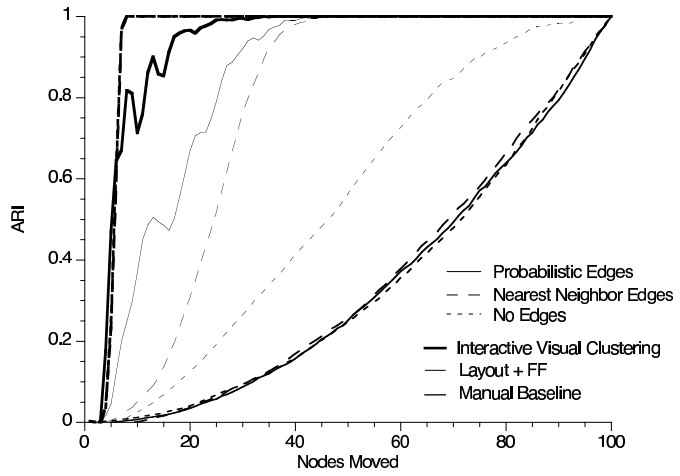

Figure 8: Effect of different types of edges on the experimental results for the Circles data set.

Circles As can be seen in Figure 7, the results on this data set are as predicted. Manually moving the instances shows the slowest improvement as a function of the number of instances moved (the lowest dashed line). The Layout Baseline (middle dashed line) shows significant improvement over the Manual Baseline. Adding the farthest-first heuristic provides yet more improvement (middle solid line). IVC performs the best; however, the clustering baseline yields nearly identical performance to IVC. We conclude that for this data set, when using clustering, the farthest-first heuristic does not provide any additional benefit. This is not surprising, since the instances are so well separated, making this a fairly easy clustering problem.

In this data set, we are also interested in understanding the effect of edge generation on performance (Figure 8). When no clustering is used, having edges increases the speed of convergence to the correct clustering. This can be seen in the middle, light-colored lines in the graph: The "No edges" version of the data set results in the worst performance, with some improvement for nearest-neighbor edges, and still more improvement for probabilistic edges.

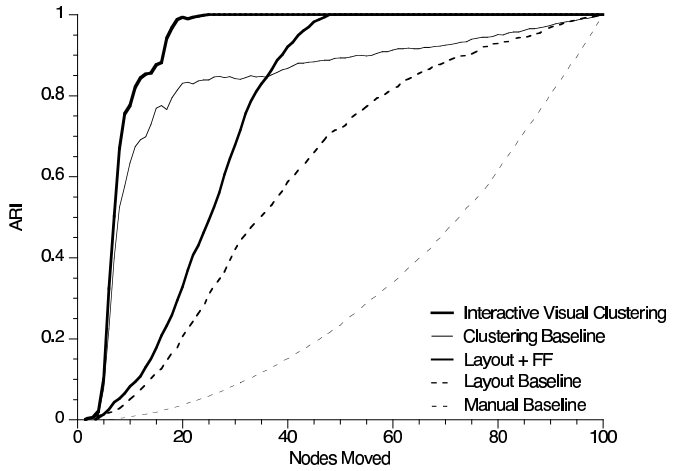

Figure 9: Experimental results on the Overlapping Circles data set.

Probabilistic edges most likely outperform nearest-neighbor edges on this data set simply because there are more edges. As a result, more instances are pulled towards a cluster when the user moves a single instance. However, when clustering is used (upper, thicker lines), the data sets with no edges or with nearest-neighbor edges perform better than the data set with probabilistic edges. The probabilistic edges on this data set contain edges between instances that do not belong to the same cluster, so the edges are only partially correlated with cluster membership. As a result, the probabilistic edges may pull some instances into the incorrect cluster. By contrast, the nearest-neighbor edges for this data set only connect instances that are in the same cluster.

Overlapping Circles Figure 9 shows the experimental results for the Overlapping Circles data set. Again, the methods perform as expected, with Interactive Visual Clustering outperforming the other methods. In this case, IVC does provide a noticeable improvement beyond the Clustering Baseline, indicating that the farthest-first heuristic is helpful in identifying important instances for repairing the clustering.

As with the Circles data set, clustering performance is worse when using the probabilistic edges, which connect instances in different clusters (Figure 10). However, for IVC, the data set using nearest-neighbor edges actually results in slightly worse performance than using no edges. This happens because the nearest-neighbor edges connect some instances that are not in the same cluster. By contrast, for Layout + FF (i.e., without clustering), using edges yields better performance than no edges. In this case, the edges do provide some useful—if noisy—information about cluster membership.

Iris As seen in Figure 11, the Interactive Visual Clustering method also yields the best performance of any of the methods we tested on the Iris data set. The improvement provided by IVC is quite noticeable in this data set: after only 10 instances, with IVC, the clusters are nearly perfect, with an

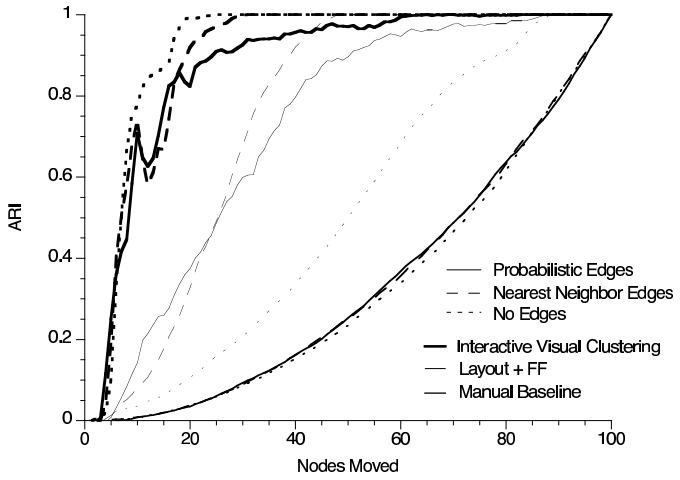

Figure 10: Effect of different types of edges on the experimental results for the Overlapping Circles data set.

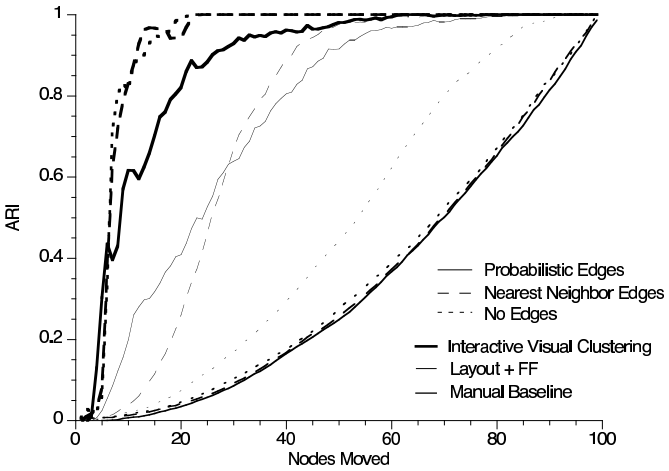

Figure 12: Effect of different types of edges on the experimental results for the Iris data set.

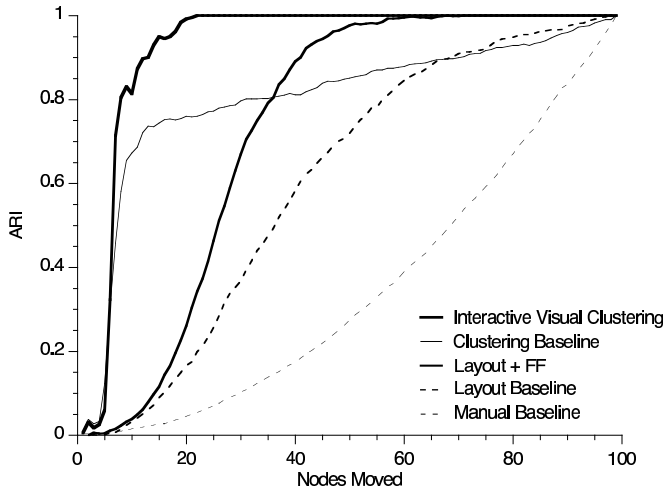

Figure 11: Experimental results on the Iris data set.

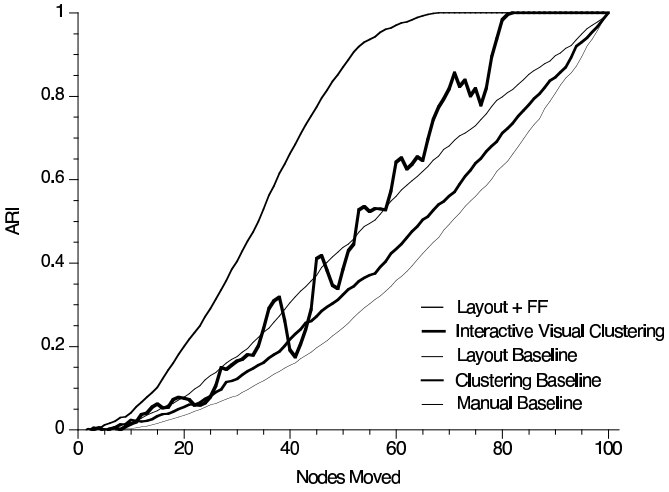

Figure 13: Experimental results on the Amino Acid Indices data set.

ARI close to 1.0. The next-best method (Clustering Baseline) has only reached an ARI of 0.75 at this point.

As in the Overlapping Clusters data set, when using clustering, probabilistic edges result in worse performance than nearest-neighbor edges (Figure 12). Again, this is likely due to the fact that the probabilistic set has more edges between instances in different clusters.

Amino Acid Indices Figure 13 shows the results for the Amino Acid Indices data set. The results on this data set are not as predicted. Clustering does not help, and actually appears to hinder performance. Here, the best performance is given by Layout + FF. IVC is only slightly better than the Clustering Baseline and the Layout Baseline. IVC also shows much more variability than the other methods: it appears that for this data set, slight variations in the layout (resulting in different selected nodes) yield significantly different clusters.

In investigating these results, we found that the clusters are not well separated in Euclidean space. Since the clustering algorithms use a Euclidean distance metric, the underlying assumptions of the clustering method are violated. However, there *is* a higher statistical correlation among the attribute values within clusters than is seen across clusters. Therefore, using a different clustering method, perhaps one based on spectral analysis [22, 16], might yield better performance. This observation also led us to develop the alternative (Amino Acid) data set.

Amino Acid The results for the Amino Acid data set are shown in Figure 14. IVC outperforms the other methods, but the Layout + FF approach is comparable. The latter method slightly outperforms IVC when only a few nodes have been moved, but IVC is slightly better for more nodes. These differences, however, are not statistically significant.

Similarly, the Clustering Baseline and Layout Baseline per-

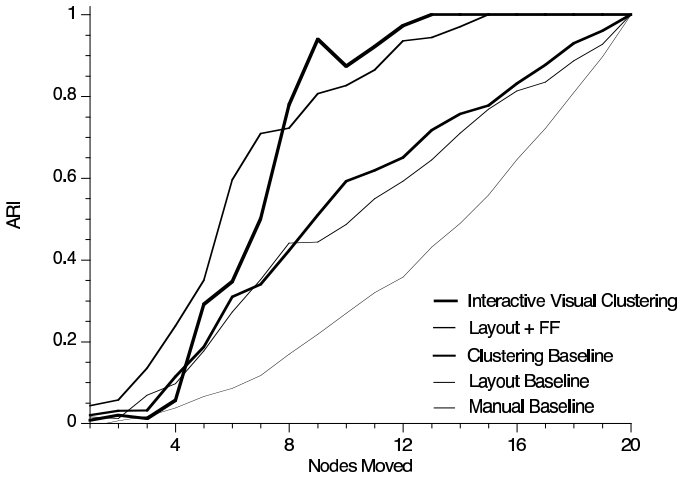

Figure 14: Experimental results on the Amino Acid data set.

form about equally, both outperforming the Manual Baseline. We conclude that the force-directed layout (taking advantage of the relational structure) and the farthest-first heuristic (identifying significant errors) help to guide the user towards the correct clustering. However, the clustering itself does not provide much, if any, additional benefit. Again, we see that in this data set, the Euclidean distances between instances not strongly related to the true clustering (although the relationship is stronger than in the Amino Acid Indices data set, thus the slightly better performance that we see here).

RELATED WORK

Lesh *et al.* [12] presented an interactive clustering method that also used force-directed layout to create the visual representation of the data. There are a few differences in the problem and approach. First, the underlying clustering method is purely graph-based, not attribute-based. Second, rather than using constrained clustering, their approach uses the modified clusters produced by the user as seeds for local heuristic search. However, their results show that similar interactive approaches may be useful even for much larger data sets than we have studied.

In the constrained clustering literature, there has been some work on active (automatic) selection of constraints [2, 23, 5]. However, we are not aware of any previous work on interactive methods for enabling the user to select appropriate constraints more effectively.

FUTURE WORK AND CONCLUSIONS

We have shown that Interactive Visual Clustering can improve clustering performance by integrating force-directed graph layout techniques with user interaction and constrained clustering. The methods we have described here are only the first step towards a more user-centered approach to clustering.

We are currently designing a user study to test the hypothesis that users will be able to identify anomalous (misplaced) instances in the display, and therefore converge more quickly to the correct clustering than without the force-directed layout. We also plan to analyze and test other models of user behavior (i.e., additional instance selection and placement heuristics). Other types of user feedback may prove to be useful, such as annotations describing why a particular instance was moved into a given cluster. Combining the user-guided approach of IVC with the system-guided methods of active constraint selection methods could result in a more mixed-initiative paradigm, where the user and the system jointly guide the clustering process.

The ultimate goal of our research is to design more integrated, interactive clustering methods for relational data sets than currently exist. The force-directed layout used in IVC incorporates the relational edges into the clustering process, but only indirectly. We are also developing relational constrained clustering algorithms, which cluster the data in attribute space and relational space simultaneously [1].

ACKNOWLEDGEMENTS

Adam Anthony, Blaz Bulka, Donald MacGlashan, Penny Rheingans.

REFERENCES

1. Adam Anthony and Marie desJardins. Open problems in relational clustering. In *ICML workshop on Open Problems in Statistical Relational Learning*, 2006.

2. S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 333–344, April 2004.

3. Indrajit Bhattacharya and Lise Getoor. Relational clustering for multi-type entity resolution. In *Proceedings of the Fourth International KDD Workshop on Multi-Relational Mining*, pages 3–12, 2005.

4. Ralf Brockenauer and Sabine Cornelsen. Drawing clusters and hierarchies. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, pages 193–227. Springer, 2001.

5. Nicolas Cebron and Michael R. Berthold. Mining of cell assay images using active semi-supervised clustering. In *Proceedings of the ICDM 2005 Workshop on Computational Intelligence in Data Mining*, pages 63–69, 2005.

6. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

7. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1988.

8. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

9. Genome Net Japan. Aaindex: Amino acid index database, 2006.

10. Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs: Methods and Models*. Springer, 2001.

11. S. Kawashima and M. Kanehisa. AAindex: Amino acid index database. *Nucleic Acids Research*, 28(1):374, 2000.

12. Neal Lesh, Joe Marks, and Maurizio Patrignani. Interactive partitioning. In *International Symposium on Graph Drawing*, pages 31–36, 2000.

13. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, 1967.

14. J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the IJCAI Text Mining and Link Analysis Workshop*, 2003.

15. D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

16. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.

17. prefuse.org. Prefuse: Interactive information visualization toolkit, 2006.

18. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

19. Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 870–878, Seattle, 2001.

20. K. Tomii and M. Kanehisa. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Engineering*, 9:27–36, 1996.

21. University of Waikato. Weka 3: Data mining with open source machine learning software in java, 2006.

22. Yair Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 975–982, 1999.

23. Qianjun Xu, Marie desJardins, and Kiri Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Proceedings of Discovery Science 2005*, 2005.

24. Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Cross-relational clustering with user's guidance. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data*, pages 344–353, 2005.