In Proceedings of the Seventh International Workshop on Agents and Data Mining Interaction (ADMI), Taipei, Taiwan, May 2011. Post-proceedings in Springer LNAI 7103.

Toward a methodology for agent-based data mining and visualization

Elizabeth Sklar[1,2], Chipp Jansen[1,3], Jonathan Chan[1] and Michael Byrd[2]

[1] Brooklyn College, The City University of New York, USA
[2] The Graduate Center, The City University of New York, USA
[3] Hunter College, The City University of New York, USA
sklar@sci.brooklyn.cuny.edu,chipp@chipp.org,jonmchan@gmail.com,mbyrd1@gmail.com

Abstract. We explore the notion of agent-based data mining and visualization as a means for exploring large, multi-dimensional data sets. In Reynolds' classic flocking algorithm (1987), individuals move in a 2-dimensional space and emulate the behavior of a flock of birds (or "boids", as Reynolds refers to them). Each individual in the simulated flock exhibits specific behaviors that dictate how it moves and how it interacts with other boids in its "neighborhood". We are interested in using this approach as a way of visualizing large multi-dimensional data sets. In particular, we are focused on data sets in which records contain time-tagged information about people (e.g., a student in an educational data set or a patient in a medical records data set). We present a system in which individuals in the data set are represented as agents, or "data boids". The flocking exhibited by our boids is driven not by observation and emulation of creatures in nature, but rather by features inherent in the data set. The visualization quickly shows separation of data boids into clusters, where members are attracted to each other by common feature values.

1 Introduction

We are motivated to explore the notion of agent-based data mining visualization, taking inspiration from the *Artificial Life* and *Information Visualization* communities. Advances in computer graphics, processor speed and networking bandwidth over last decade have made possible the application of dynamic techniques for information visualization that were previously limited to high-end graphics laboratory settings. In addition, the rapid rise in popularity of certain types of data visualization environments, particularly those from the *Geographic Information Systems (GIS)* community, have made commonplace otherwise obscure techniques for examining vast multi-dimensional data sets. *Google Earth* [2, 6, 8] is one example, which has brought to the masses the notion of zoomable interfaces and allowed everyday computer users to explore 3-dimensional geographic data sets facilitated by, what are now considered to be, standardized

controls. Our work takes advantage of these trends by hypothesizing that when any multi-dimensional data set is projected onto 2 or 3 dimensions, it can be explored as if it were a geographic landscape. Such "data landscapes" could be studied dynamically in interesting ways if points in the landscape are represented as software agents that move in response to various stimuli.

The work presented here develops this idea, with the long term goal of providing a participatory agent-based data mining and visualization system in which the user and the system collaboratively explore a data landscape. Classical statistics are often fine (and the right choice) when the user has some notion of what is in their data set and how they want to analyze their data. Looking for a bell curve in a data set of student grades is an example where a standard statistical method, like computing mean and standard deviation, is an appropriate choice. Using K-means [9] when clustering data into a known number of groups is an example where a standard machine learning method is appropriate. But when the number of clusters and even the selection of features on which to cluster the data are unknown *a priori*, other techniques must be investigated. Our hypothesis is that a *participatory* system in this type of situation can take advantage of superior human skills for quick visual understanding and intuition, facilitating a user to easily identify and nudge an evolutionary data mining process into a desired direction.

This paper is organized as follows. Section 2 describes the related work in this area, which was inspired by seminal work on "flocking boids" [16]. Sections 3 details our approach. Section 4 presents results from applying our approach to a sample data set. Finally, we conclude in Section 5 with summary remarks.

2 Related Work

In 1987, Cliff Reynolds [16] produced the classic work on flocking in artificial systems. Reynolds' model focuses on graphical aspects, and the aim was to produce a realistic (from a graphical standpoint) simulation of a group of identical agents (which Reynolds calls "boids"). Each agent is given the same behavioral rules, which includes instructions for how to react to others in an agent's "neighborhood". These interaction instructions consist of three independent rules. First, there is a *separation* rule, which implements collision avoidance, preventing the agents from bumping into each other. Second, there is an *alignment* rule, which causes agents to move at the same velocity and heading as those around them. Third, there is a *cohesion* rule, which encourages agents to gravitate towards a common center. When put together, the composite set of agents exhibits emergent group behavior that looks like *flocking*—in the same way that schools of fish or flocks of birds or herds of land animals move together. This work has proven to be highly influential and has inspired most of the subsequent research in the area of artificial flocking.

Proctor and Winter [15] developed the notion of *information flocking*, about 10 years after Reynolds' work. Their aim was to visualize patterns of behavior of users visiting web sites. They simulated artificial "fish" and associated a fish

with each user, mining users' clickstreams to provide input to their simulation. A user clicking on a URL was interpreted as interest in the topic(s) displayed on the page. A matrix of user interests was updated, and the fish responded by grouping together—showing users who shared similar interests.

Moere [11] provides a comprehensive overview of decentralized data visualization work conducted in the two decades since Reynolds' original work. He divides the work in the field into three categories: *information particle* animation; *information flocking*; and *cellular ant* methods. The first category involves simulating a group of *information particles*, or *"infoticles"* (i.e., agents), in three-dimensional space. Agents react to *forces* in the system, moving in response to them. The forces can emanate from static points in the agents' artificial landscape, acting as fixed point attractors (or repellers), as well as from dynamic points—other agents. The forces influence the velocity and direction of movement of each agent. With an infoticle, data values are assigned to each agent. The forces acting on an agent are calculated according to the similarity (or dissimilarity) of the agents' data values in relation to those of other nearby agents or fixed-point attractors in the landscape.

The second category, *information flocking*, describes the method introduced by Proctor and Winter [15], discussed above. Picarougne *et al.* [12] extend the work of Proctor and Winter by introducing the notion of an "ideal" distance between agents that is proportional to the difference in feature values between the agents. They compare to the K-means clustering method and state that their results are similar. Another example that extends Proctor and Winter's work suggests using the visualization in tandem with other algorithms for data classification [1]. Moere [10] offers a nice extension to the Proctor and Winter work by applying the method to time-varying datasets and using financial stock market data as an example. The data values (e.g., prices or "quotes" of a given stock) are represented by agents, and the agents' values change over time. Moere introduces two additional behavior rules: *data similarity*, where agents stay close to others with similar data values; and *data dissimilarity*, where agents move away from others with different data values. The method highlights changes in "data behavior"—as data values change over time, agents that have similarly changing data values end up clustering together.

The third category combines ideas from *artificial ant systems* [4], *ant foraging* [3] and *cellular automata* [19]. The general idea is that artificial agents ("ants") move around a two-dimensional grid world and collect "food"—i.e., data. The aim is to bring like pieces of data together, emulating the way that ants gather food particles and deposit them in shared nests. The agents' movement is constrained by cellular-automata-like rules in which they respond only to neighboring grid cells. While an interesting alternative, many comment that this approach takes more time to reach a solution than other methods. The reason is that the agents' movements are constrained (by the cellular automata rules), and so even if an agent "knows" where it is going, it still has to find a path to get there. The advantage, however, is that forcing the agent to explore an indirect path will lead it to encounter other agents and new data (food) sources, which

may eventually result in a more robust solution (even if it takes longer to find). Handl and Meyer [7] review ant-based clustering algorithms, comparing results of different implementations and discussing approaches from an optimization perspective.

Cui *et al.* [20] employ a flocking-based algorithm for document clustering analysis. Each document object is represented as a "boid" and projects the document's TFIDF[4] vector onto a two-dimensional space. The authors compare their flocking-based clustering algorithm with K-means clustering and Ant clustering. They found that the flocking algorithm ran faster than the Ant algorithm, while K-means ran the fastest. However, the K-means method requires an *a priori* estimate of the correct number of groups, whereas the flocking and Ant-based methods do not. This makes the flocking algorithm the better choice for a dynamically changing data set, like the one used by the authors.

Picarougne *et al.* [13] describe a biologically inspired clustering algorithm, called *FClust*, which applies similar, but not identical, techniques to those of Proctor and Winter. The authors also model agents as representations of data points moving in a two-dimensional landscape. However, instead of applying three forces to the agents' motion, the authors apply two forces: one that attracts agents with similar data values and one that repels agents with dissimilar data values. They determine experimentally the threshold for similarity, showing visually that different threshold values produce different clustering results, as one would expect. They apply their method to several data sets and compare results to K-means clustering methods. They also describe a component of their system referred to as "interactive clustering" in which a user can select and label data elements on a visual display and the system will perform clustering on the user-selected classes. The authors define an "entropy" measure (inspired by Shannon [17]) for evaluating the stopping condition for the system.

The work we present in the remaining sections has been inspired and informed by these and other examples, all of which were found primarily in the *Artificial Life* and *Information Visualization* literature. Our approach is most similar to that of Moere [10] and Picarougne *et al.* [13]. However, we take an *agent-based* perspective. Each agent in our system can only compare its data values to others in its geographic neighborhood, much as a robot in a multi-robot team can only respond to teammates that are within range of its visual or range sensors or communication network. Where Moere and Picarougne *et al.* stress two behavior rules (essentially data similarity and dissimilarity), we stick with Reynolds' original three behavior rules (separation, cohesion and alignment); and we introduce an additional *meta-level* flocking step in which groups that are similar emit an attractive force and tend to move towards each other. The details are described in the next section.

[4] Term Frequency, Inverse Document Frequency—a common metric used in Natural Language Processing (NLP)

3 Approach

Our approach is built on the *information flocking* paradigm described in the previous section, with modifications highlighted below. We associate a "data boid", or agent, with each record in an n-dimensional data set (i.e., a data set with n features in each record). In our visual environment, the agent moves around a two-dimensional geographic landscape, and its position is influenced by the relative values of its data features compared with those of its neighbors. Iterating over a number time steps, or "frames", we animate the agents by first applying the three standard flocking rules, then *grouping* agents based on their feature value and geographic proximity, next executing a *meta-flocking* step that pulls similar groups together, and finally applying a *braking* factor (described below) as clusters of like agents converge.

The standard flocking procedure involves computing vectors representing three forces acting on each agent, followed by combining the three forces using a weighting scheme. The three force vectors are computed as follows:

- **Separation.** A steering vector is computed that points away from agents in the neighborhood with dissimilar feature values:

$$sep = \frac{1}{n} \sum_{i=1}^{n} d_i (P - P_i) \tag{1}$$

  where n is the number of agents in the neighborhood within a specified feature distance (Δ), d_i is the geographic distance to neighbor i, P is the (x, y) position of the agent performing the computation, and P_i is the (x, y) position of neighbor i.

- **Cohesion.** A steering vector is computed that points toward the center of the group of agents in the neighborhood with similar feature values:

$$coh = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{2}$$

  where n is the number of agents in the neighborhood within a specified feature distance (Δ) and P_i is the (x, y) position of neighbor i.

- **Alignment.** A velocity vector is computed to match the average speed and heading of agents in the neighborhood with similar feature values:

$$ali = \frac{1}{n} \sum_{i=1}^{n} V_i \tag{3}$$

  where n is the number of neighboring agents within a specified feature distance (Δ) and V_i is the velocity of neighbor i.

The vectors are weighted and summed to update the agent's position:

$$velocity = \psi \cdot sep + \alpha \cdot ali + \gamma \cdot coh \tag{4}$$

The weighting scheme amongst the three vectors is important, and there is a delicate balance between them. For the results presented here, we used weights of $\psi = 1.5$, $\alpha = 1.0$ and $\gamma = 1.0$.

The standard flocking algorithm only considers geographic distance (d) between agents, whereas we use geographic distance to determine an agent's neighborhood and distance in feature space (Δ) to determine whether agents should be attracted to or repel their neighbors. The distance in feature space is computed as follows. Multi-featured data sets frequently contain a mixture of *categorical* and *quantitative* types of features. We compute the distance between individual features, based on their type and normalized to $[0 \ldots 1]$ (where 0 is most similar and 1 is most dissimilar), and then calculate the average over all features. The distance between two categorical feature values is:

$$\delta(a_i, b_i) = (a_i == b_i \ ? \ 0 : 1) \tag{5}$$

where a_i and b_i represent the values of the i-th feature for agents a and b, respectively. The distance between two quantitative feature values is:

$$\delta(a_i, b_i) = \frac{|a_i - b_i|}{max_i} \tag{6}$$

where max_i is the range of possible values for feature i. The overall feature distance between two agents is thus:

$$\Delta(a, b) = \frac{\sum_{i=1}^{n} \delta(a_i, b_i)}{n} \tag{7}$$

After the standard flocking rules are applied to all the agents, a *grouping* step takes place, followed by a *meta-flocking* step. Grouping is an organizational step in which the agents are partitioned into virtual groups based on their locations in geographic space. This is done using a recursive algorithm in which each agent looks at its neighbors (other agents within a fixed physical distance), and those neighbors look at their neighbors, and so on, adding to the group all neighbors (and neighbors' neighbors, etc.). Agents' positions do not change during the grouping step.

The meta-flocking step pulls together groups that have similar feature values. For each group, a set of feature values is calculated that represents the center of the group in feature space. For quantitative feature values, the center is the mean over all group members. For categorical feature values, the center is the mode (i.e., most popular) over all group members. A pairwise comparison is made between all groups' feature centers. A cohesion steering vector for the group is computed that points toward the geographic center of mass of other groups with similar feature values:

$$coh_g = \frac{1}{n} \sum_{i=1}^{n} M_i \tag{8}$$

where n is the number of groups within a specified feature distance (Δ) and M_i is the (x, y) position of the center of mass of group i. Note that this is different from the standard (agent-level) cohesion rule for two reasons. First, it only takes into account the feature distance between groups and does not use the geographic distance. Second, it compares with every group in the system, not just groups that are within close graphic distance proximity.

Finally a *braking* factor is applied to all members of groups whose feature values within the group are similar. As group membership settles, the system converges and the agents belonging to settled groups move more slowly and eventually halt.

As the clusters of agents evolve, the system computes two metrics to assess how the process is performing:

- *pctGood* computes the percentage of groups that fall within a pre-specified "goodness" threshold. The aim is for this value to converge to 1.0. The *goodness* of a group is a measure of the variance in group members' feature values. A goodness value close (or equal) to 0 indicates that the group members are closely matched; a higher goodness value indicates a more diverse group. Goodness is calculated as:

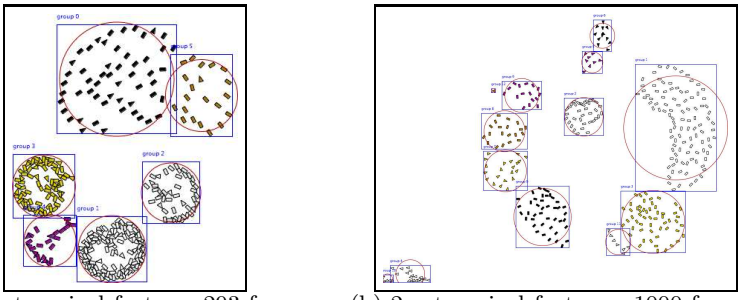
$$goodness = \sum_{i=1}^{n} \left(\sum_{f \in Q} \sigma(f) + \sum_{f \in C} (1 - \frac{match(f)}{n}) \right) \tag{9}$$

  where n is the number of agents in the group, Q is the set of each agents' quantitative features, C is the set of each agents' categorical features, $\sigma(f)$ is the standard deviation of (quantitative) feature f across all agents in the group, and $match(f)$ is the number of agents in the group with (categorical) feature f matching the mode for that feature in the group.

- *pctOverlap* computes the percentage of groups that have overlapping feature values. Since the system determines group membership dynamically, based on which agents have come within small graphical distances of each other while they float around in space, there is no guarantee that multiple groups with the same feature values will not appear. The overall aim is to find the minimal set of groups that have different feature values, since it is not desirable to have different groups that overlap in feature space. The aim is for this value to converge to 0.0.

4 Experiments and Results

We implemented our approach in a prototype system, developed using Processing [14]. The system can run in an interactive mode, where the user selects features on which to cluster, and the system responds in real-time. The user can change feature selection during clustering, and the system will adjust. To assess the efficacy of our method for clustering, we ran a series of experiments in



(a) 1 categorical feature; 293 frames
$pctGood = 1.0$, $pctOverlap = 0.0$

(b) 2 categorical features; 1000 frames
$pctGood = 0.92$, $pctOverlap = 0.08$

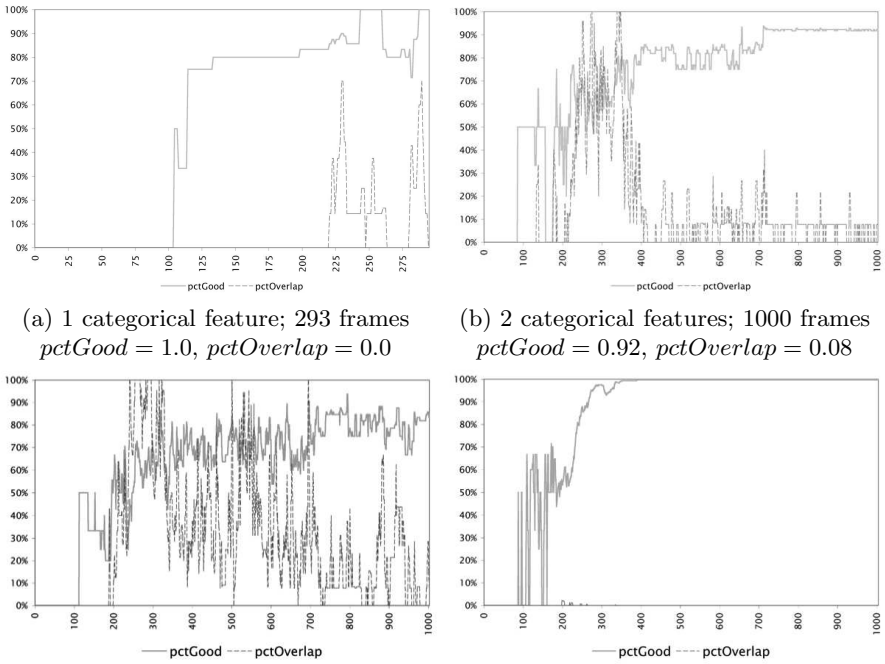
(c) 1 categorical, 1 quantitative; 1000 fr.
$pctGood = 0.83$, $pctOverlap = 0.08$

(d) all 5 features; 1000 frames
$pctGood = 0.99$, $pctOverlap = 0.0$

Fig. 1. Screen shots at end of 4 different runs, each clustering on different sets of features, as indicated above.

a batch mode, using a 5-dimensional sample set of university enrollment data, gathered over a 4-and-a-half-year period. The data was partitioned into subsets, each representing one term; on average, there are 375 data records per term. Each record represents an instance of a student taking one class in a specific term. The fields in each record are: a unique identifier for the student (categorical feature), a course number identifying which class was taken (categorical feature), the student's gender (categorical feature), a numeric code indicating the student's ethnic origin (categorical feature), and the grade earned by the student in the class (quantitative feature). A subset of data for one term contains all the students who took classes in that term. A student who took more than one class in a given term is represented by multiple data records.

Figure 1 contains screen shots from 4 representative runs of the system, captured at the end of each run. The end of a run is either the point at which $pctGood = 1.0$ and $pctOverlap = 0.0$ or a maximum number of frames have elapsed. We used 1000 as the maximum number of frames for the runs presented here. The figure captions show the number of frames elapsed for each run, and the final values of *pctGood* and *pctOverlap*. Figure 2 shows the improvement in *pctGood* and *pctOverlap* rates through the course of each corresponding run. When clustering on one categorical feature value, the system quickly converges



(a) 1 categorical feature; 293 frames
$pctGood = 1.0$, $pctOverlap = 0.0$

(b) 2 categorical features; 1000 frames
$pctGood = 0.92$, $pctOverlap = 0.08$

(c) 1 categorical, 1 quantitative; 1000 fr.
$pctGood = 0.83$, $pctOverlap = 0.08$

(d) all 5 features; 1000 frames
$pctGood = 0.99$, $pctOverlap = 0.0$

Fig. 2. Improvement in evolution metrics: *pctGood* and *pctOverlap*

to the correct solution (in 293 frames); whereas in all the other cases, the system timed out after 100 runs. Even so, the percentage of "good" groups was high, ranging between 83% and 99%.

At the end of each run, we computed two scores to determine how well the clustering process worked: *within cluster* score and *between cluster* score. The "within cluster" score measures the disparity of the feature values in the cluster. The goal is to minimize the "within" cluster score and to maximize the "between" cluster score. The "within" cluster score is determined by calculating the feature distance (equation 7) from the representative center of the group in feature space (see the description of the meta-flocking step in Section 3) to each member of the group. The "between" cluster score is determined by calculating the average over all pairwise feature distances between representative (feature) centers of each group. Using these two metrics, we compared our results to two standard clustering algorithms from the data mining literature: K-means [9] and *Cobweb* [5]. We ran these algorithms using WEKA [18] on our same data set.

Figures 3 and 4 compare the results of our agent-based data visualization method to K-means and Cobweb. Each vertical bar represents an average over

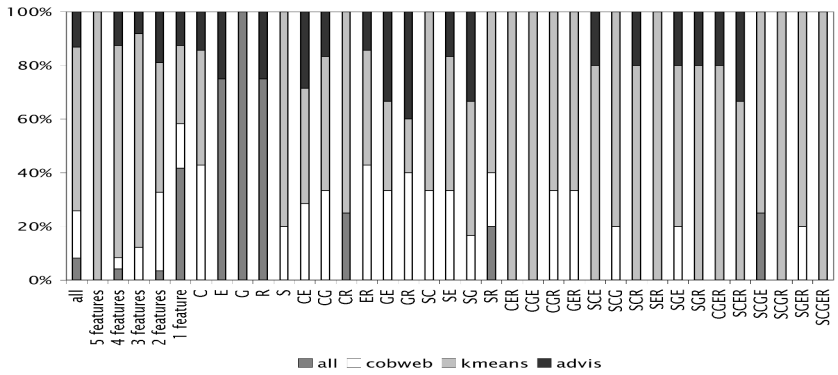


Fig. 3. "Within Cluster" score comparison

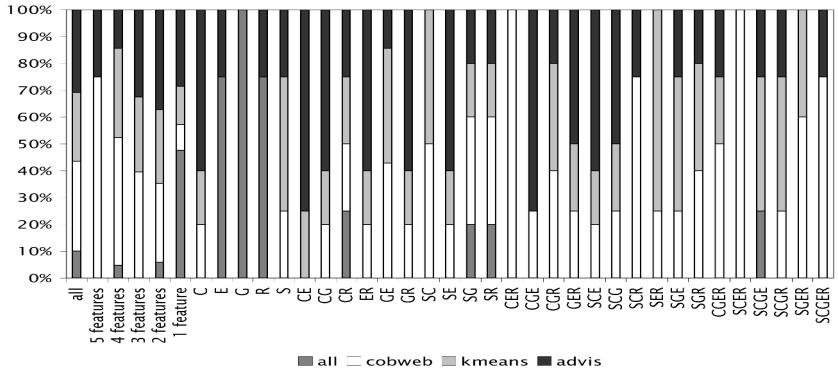


Fig. 4. "Between Cluster" score comparison

multiple runs[5], clustering on different combinations of features. The leftmost bar shows the average results over all combinations of features. The next 5 bars show the average results when clustering over 5, 4, 3, 2, and 1 feature(s) (in aggregate, regardless of which particular features). The remaining 31 bars show average results when clustering over specific combinations of all the various features. The shaded bands within each bar represent the distribution (by percentage) of best results for each clustering method. In general, K-means produces the best results for "within" cluster score more frequently than the other methods. For "between" cluster score, the results are split between Cobweb and our agent-based data flocking method. Table 1 quantifies the error rate for each method when it was not categorized as the "best" (see table caption for an explanation

[5] The number of runs varied between 4 and 31, depending on the number of features being clustered.

	K-means	cobweb	advis
within cluster	**0.87%**	38.23%	47.39%
between cluster	1.58%	2.18%	**1.12%**

Table 1. The values in the table are are computed as follows: for each method (shown in columns), when it did not produce the best score, the error is calculated as the difference between that method's score and the best score; this error is expressed as a percentage of the best score in order to give a sense of the proportion of the error.

of how the error rate is computed). K-means produces the smallest error rate for "within" cluster score, whereas our method (labeled "advis") produces the smallest error rate for "between" cluster score.

5 Summary

We have described preliminary work in the development of an agent-based data mining and visualization method for clustering multi-dimensional data sets. Our technique extends early work in the area of information flocking and introduces several strategies that help the system converge on a clustering task. The first strategy involves "grouping" and "meta-level" flocking steps in which groups of data boids are identified and nudged toward each other, based on the similarity between feature values amongst the groups. The second strategy is a "braking" adjustment that causes the data boids to decrease their velocity as the system converges on good clusters. Experiments were conducted on a sample data set, showing promising results for "between cluster" scoring as compared with standard techniques. The advantage that our method has over standard techniques is the ability to adapt during clustering to changes in clustering criteria.

We plan to apply evolutionary machine learning techniques to evaluate various parameter settings, including weights (equation 4), "goodness" threshold and geographic neighborhood distance, in order to improve clustering results. Eventually, the aim is for the system to dynamically explore various combinations of features while clustering, learning to converge on the set of features that offer the best cluster scores. In addition, we also will apply our method to real-time data streams, where the feature values in the data set change while the clustering process is occurring. Because our method is highly dynamic, it should be able to respond in near real-time (depending on the relative speed at which new data comes in compared to the time it takes clusters to form).

6 Acknowledgments

We would like to thank the anonymous reviewers of this paper for their comments, which were not only helpful for the current revision but also provide guidance for the next stages of our work. This project was supported in part by the National Science Foundation (#CNS-0722177).

References

1. Aupetit, S., Monmarché, N., Slimane, M., Guinot, C., Venturini, G.: Clustering and dynamic data visualization with artificial flying insect. In: Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation (GECCO), Lecture Notes In Computer Science. pp. 140–141. Springer-Verlag (2003)
2. Butler, D.: Virtual globes: The web-wide world. Nature 439, 776–778 (2006)
3. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétian, L.: The dynamics of collective sorting: Robot-like ants and ant-like robots. In: From Animals to Animats: 1st International Conference on Simulation of Adaptative Behaviour. pp. 356–363 (1990)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics-Part B 26(1), 1–13 (1996)
5. Fisher, D.H.: Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning 2, 139–172 (1987)
6. Google: Google earth. `http://earth.google.com` (2005)
7. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
8. Lisle, R.J.: Google earth: a new geological resource. Geology Today (2006)
9. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. pp. 281–297 (1967)
10. Moere, A.V.: Time-varying data visualization using information flocking boids. In: Proceedings of IEEE Symposium on Information Visualization. pp. 10–12 (2004)
11. Moere, A.V.: A model for self-organizing data visualization using decentralized multiagent systems. In: Prokopenko, M. (ed.) Advances in Applied Self-organizing Systems, Advanced Information and Knowledge Processing, vol. Part III, pp. 291–324. Springer (2008)
12. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: On data clustering with a flock of artificial agents. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI). pp. 777–778 (2004)
13. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: A new approach of data clustering using a flock of agents. Evolutionary Computation 15(3), 345–367 (2007)
14. Processing: `http://www.processing.org/` (2010)
15. Proctor, G., Winter, C.: Information flocking: Data visualisation in virtual worlds using emergent behaviours. In: Proceedings of Virtual Worlds. pp. 168–176. Springer-Verlag (1998)
16. Reynolds, C.W.: Flocks, Herds and Schools: A Distributed Behavioral Model. In: International Conference on Computer Graphics and Interactive Systems. pp. 25–34 (1987)
17. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal 27, 379–423 (1948)
18. WEKA: `http://www.cs.waikato.ac.nz/ml/weka/` (2010)
19. Wolfram, S.: Cellular automata as models of complexity. Nature 311, 419–424 (1984)
20. Xiaohui Cui, J.G., Potok, T.E.: A flocking based algorithm for document clustering analysis. Journal of Systems Architecture 52(8-9), 505–515 (2006)