A Distributed Flocking Approach for Information Stream Clustering Analysis

Xiaohui Cui and Thomas E. Potok
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6085
Cuix, potokte@ornl.gov

Abstract

Intelligence analysts are currently overwhelmed with the amount of information streams generated everyday. There is a lack of comprehensive tool that can real-time analyze the information streams. Document clustering analysis plays an important role in improving the accuracy of information retrieval. However, most clustering technologies can only be applied for analyzing the static document collection because they normally require a large amount of computation resource and long time to get accurate result. It is very difficult to cluster a dynamic changed text information streams on an individual computer. Our early research has resulted in a dynamic reactive flock clustering algorithm which can continually refine the clustering result and quickly react to the change of document contents. This character makes the algorithm suitable for cluster analyzing dynamic changed document information, such as text information stream. Because of the decentralized character of this algorithm, a distributed approach is a very natural way to increase the clustering speed of the algorithm. In this paper, we present a distributed multi-agent flocking approach for the text information stream clustering and discuss the decentralized architectures and communication schemes for load balance and status information synchronization in this approach.

1. Introduction

Clustering analysis is a descriptive data mining task, which involves dividing a set of objects into a number of clusters. The motivation behind clustering a set of data is to find inherent structure in the data and expose this structure as a set of groups [1]. The data objects within each group should exhibit a large degree of similarity while the similarity among different clusters need be minimal [8]. Document clustering is a fundamental operation used in unsupervised document organization, automatic topic extraction and information retrieval. It provides a structure for organizing a large body of text for efficient browsing and searching. Researches in document clustering analysis mainly focus on how to quickly and accurately cluster static document collection. Research on clustering the dynamic text information stream is limited. However, currently there is an increasing needed for clustering analysis of the dynamic documents to meet the challenge of analyzing text information stream generated everyday.

New algorithms based on biological models, such as ant colonies, bird flocks, and swarm of bees etc., have been invented to solve problems in the field of computer science. These algorithms are characterized by the interaction of a large number of agents that follow the same rules. The Flocking model is one of the first collective behavior models that have been applied in popular applications, such as animation. In addition to being used to simulate group motion, which has been used in a number of movies and games, Flocking model has already inspired researches in time varying data visualization [11, 12] and spatial cluster retrieval [5].

Early research [3] has resulted in a flocking based clustering algorithm that can achieves better performance than the K-means and the Ant clustering algorithm in document clustering. This algorithm generates clusters of a given set of data through embedding high dimensional data items on a two-dimensional grid for easy clustering result retrieval and visualization. The heuristic algorithm can continually refine the clustering result and quickly react to the change of individual data. This character makes the algorithm suitable for clustering dynamic changed document information, such as the text information stream. In this paper, we propose a bio-inspired clustering model, the Multiple Species Flocking clustering model (MSF), and a distributed multi-agent

MSF approach for dynamic updated text information stream clustering.

The remainder of this paper is organized as follows: Section 2 briefly discusses the related works in the traditional and bio-inspired document clustering area. Section 3 proposes a new multiple species flocking (MSF) model to model the multiple species bird flock behaviors and a MSF clustering algorithm for document clustering. The Multi-Agent MSF approach for Distributed Dynamic Document Clustering is presented in section 4. Section 5 provides the detailed experimental setup and results in comparing the performance of the multi-agent implementation for clustering the dynamic updated document stream on the cluster computer and a single processor computer. The conclusion is in Section 6.

2. Related Works

There are two major clustering techniques: partitioning and hierarchical [8]. Most document clustering algorithms can be classified into these two groups. In recent years, it has been recognized that the partitioning techniques are well suited for clustering a large document dataset due to their relatively low computational requirements [15]. The best-known partitioning algorithm is the K-means algorithm and its variants [7, 14]. This algorithm is simple, straightforward and based on the firm foundation of analysis of variances. One drawback of the K-means algorithm is that the clustering result is sensitive to the selection of the initial cluster centroids and may converge to the local optima. The other limitation of the K-means algorithm is that it generally requires a prior knowledge of the probable number of clusters for a document collection.

To deal with the limitations that exist in the traditional partition clustering methods, a number of computer scientists have proposed several approaches inspired from biological collective behaviors to solve the clustering problem, such as Genetic Algorithm (GA) [9], Particle Swarm Optimization (PSO) [2, 10], Ant clustering [6] and Self-Organizing Maps (SOM) [16]. Within these clustering algorithms, Ant clustering algorithm is a partitioning algorithm that does not require a prior knowledge of the probable number to clusters or the initial partition. Wu [17] and Handl [6] proposed the use of Ant clustering algorithms for document clustering and declared that the clustering results from their experiments are much better than that from K-means algorithm. However, in the Ant clustering algorithm, the clustered data objects do not have mobility themselves. The movement of data objects has to be implemented through the movements of a small number of ant agents, which will slow down the clustering speed. Since the ant agent that carries an isolated data object does not communicate with other ant agents, it does not know the best location to drop the data object. The ant agent has to move or jump randomly in the grid space until it finds a place that satisfies its object dropping criteria, which usually consumes a large amount of computation time. Our experiment results show that the Ant clustering algorithm needs more iteration to generate an acceptable clustering result.

3. The Multiple Species Flocking (MSF) Clustering Algorithm

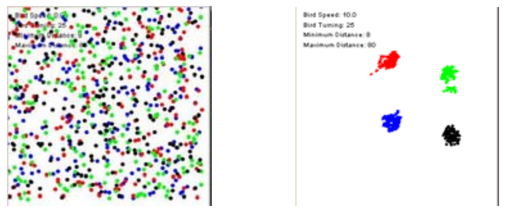
3.1 Flocking Model

Flocking model was first proposed by Craig Reynolds [13]. It is a bio-inspired computational model for simulating the animation of a flock of entities called "boid". It consists of three simple steering rules that need to be executed at each instance over time. Three basic rules include: (1) Separation: Steering to avoid collision with other boids nearby. (2) Alignment: Steering toward the average heading and match the velocity of the neighbor flock mates. (3) Cohesion: Steering to the average position of the neighbor flock mates. The three basic rules are sufficient to reproduce the moving behaviors of a single species bird flock on the computer. However, our experiments indicate these three rules will eventually result in all boids in the simulation forming a single flock. It can not reproduce the real phenomena in the nature: the birds or other herd animals not only keep themselves within a flock that is composed of the same species or the same colony creatures, but also keep two or multiple different species or colony flocks separated.

3.2 Multiple Species Flocking Model

In this report, we propose a new Multiple Species Flocking (MSF) model to model the multiple species bird flock behaviors. In the MSF model, in addition to the three basic action rules in the Flocking model, a fourth rule, "feature similarity rule", is added into the basic action rules of each boids to influence the motion of the boids. Based on this rule, the flock boid tries to stay close to other boids that have similar features and stay away from other boids that have dissimilar features. The strength of the attracting force for similarity boid and repulsion force for dissimilarity boid is inversely proportional to the distance between the boids and the similarity value between the boids' features.

Based on the MSF model, we implemented the multiple species bird flock simulation as shown in Figure 1. In this simulation, there are four kinds of boid species and each species have 200 boids. We use four different colors to represent different species. All together, 800 boids are simulated in the environment. At the initial stage, each boid is randomly deployed in the environment as shown in Figure 1(a). Each color dot represents one boid. There is no central controller in the simulation. Each boid can only sense boids within a limited range and move in the simulation environment by following the four action rules of MSF model. Although there is no intention for each boid to form a same species group and to separate the different species from each other, after several iterations, as shown in Figure 1(b), the boids in same species are grouped together and different species are separated.



(a) (b)

Figure 1: Multiple species bird flocking simulation

3.3 MSF document clustering algorithm

One application of the MSF model is the clustering document collection [3]. Inspired by the bird's ability of maintaining a flock as well as separating different species or colony flocks, the MSF clustering algorithm uses a simple and heuristic way to cluster document datasets. In the MSF clustering algorithm, we assume each document vector is projected as a boid in a 2D virtual space. The document vector is represented as the feature of the boid. The boids that share similar document vector feature (same as the bird's species and colony in nature) will automatically group together and became a boid flock. Other boids that have different document vector features will stay away from this flock. After several iterations, the simple local rules followed by each boid results in generating complex global behaviors of the entire document flock, and eventually a document clustering result is emerged.

4. Multi-Agent Approach for Document Clustering

Inevitably, the MSF clustering algorithm approach of using single processor machine to cluster the dynamic text stream requires a large amount of memory and faster execution CPU. Since the decentralized character of the MSF algorithm, using Multi-Agent techniques to develop a distributed MSF clustering approach can increase the clustering speed of the algorithm.

In the MSFC algorithm, the document parse, similarity measure and boid moving velocity calculation are the most computational consumption parts. The distributed implementation should divide these computational tasks into smaller pieces that can be scheduled to concurrently run on multiple processors. In order to achieve good performance on distributed computing, several issues must be examined carefully when designing a distributed solution. First is the load balance. It is important to keep load balancing among processing nodes to make sure each node have approximately same workload. Second is the environment states synchronization. It is necessary for a distributed implementation to develop a synchronization algorithm which is capable of maintaining causality. Third is reducing the communication between nodes. That include the communication overhead of the environment states synchronization and control message exchange between nodes.

Based on these requirements, we developed a distributed multi-agent based (MAB) implementation of the MSF clustering algorithm for clustering the text information stream. In MAB, each boids are modeled and implemented in terms of software agents, which makes boids pro-active, adaptive and communicable. The MAB implementation supports distributed load balance in a very natural way. Since each boid agent is implemented to work for document retrieval, parse, similarity comparison and moving velocity calculation independently, it is straight-forward to let different agents run on different machines to achieve the load balance. Each agent can be added, removed or mobiled to other machine without interrupt other agent's running. The system can be scalability and pro-activity to the change of work load. In the following sessions, we described the two experiments that helped us to find the best schemes for load balancing, synchronizing and agent communicating in the MSF multi-agent approach.

5. Experiments and Results

5.1 Multi-Agent platform and experiment environment setup

The distributed MSF clustering algorithm is implemented on the JADE agent platform. JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. JADE is a FIPA compliant agent platform. As a distributed agent plate form, the JADE agent can be split on several hosts. The OS on each host is not necessary same. The only environment required is a Java virtual machine (JVM). Each JVM is a basic container of agents that provide a complete run time environment for agent and allow several agents to concurrently execute on same container.

In the MSF MAB implementation, each boid is implemented as a jade agent. The agent has the ability to calculate moving velocity based on the four actions rules we discussed in previous session. Each agent carries a feature vector to represent a document vector. The environment used in the experiment consists of a continuous 2D plane, in which agents are placed randomly on a grid within a 4000×4000 squire unit area. All experiments were carried out on an experiment Linux computer cluster machine. The cluster machine consists of one head node, ASER and three cluster nodes, ASER1, ASER2, and ASER3, which are connected with Gigabit Ethernet switch. Each node contains one 2.4G Intel Pentium IV processor and 512M memory. To graphically display the usage of CPU and network bandwidth of each node in the computer cluster, we used Linux cluster management software, LCM (http://linuxcm.sourceforge.net/), for real-time displaying all cluster nodes' processor and network usage.

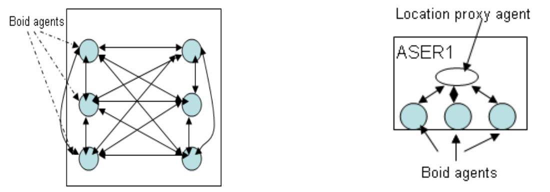
5.2 Datasets

The document dataset used in the experiments is derived from the TREC-5, TREC-6, and TREC-7 collections and represented as a set of vectors $X=\{x_1, x_2, \ldots, x_n\}$, where the vector x_i corresponds to a single object and is called *"feature vector"* that contains proper features to represent the object. The feature value is represented using the Vector Space Model (VSM) [4]. Before translating the document collection into VSM, the very common words (e.g. function words: "a", "the", "in", "to"; pronouns: "I", "he", "she", "it") are stripped out completely and different forms of a word are reduced to one canonical form by using Porter's algorithm [12].

To simulate the dynamic updated document collection, the document vector of each agent is periodically updated with new document vector and old document vector is considered as expired. For easy comparing the performance of different scenario, in the experiments, each agent's document feature will be updated by new document for ten times during the whole life of the system execution. In each experiment, the system will run 1000 cycles and the average document update gap is 100 time-steps.

5.3 Experiment 1: Communication schemes

In this experiment, we compared the performance of different communication schemes for boid agents exchanging the environment information when flying in the virtual space. In a distributed system, the environment information is spread out among the processors involved in the system. An agent doesn't know other agent information if it is not informed, it has to commute with other agents to collect enough information, does an exhaustive search to find out which agents are located within its range, and calculates the force that it is pushed to travel based on the neighbor agents' information. All these require that each agent in the system have a global view of other agents' status information.
There are two basic communication schemes to update the agent's information on different processors. The easy communication scheme is broadcast. As shown in Figure 2(a), each agent in the system broadcast its status information to all other agents wherever they are located in the same node or different nodes. Each agent will also use the information it received from other agents' broadcast to find out its neighbor boid mates and calculate the next moving velocity.



(a) Broadcast (b) Location Proxy

Figure 2: The architectures of different communication schemes

Another scheme is the location proxy scheme. As shown in Figure 2(b), there is a location proxy agent in the computer node. Each agent will only inform its status to the location proxy agent in the same node. The agent also inquires the location proxy agent to find out its neighbor mates. At every time step, after collecting the status of all agents that located in the same node, location proxy agents will broadcast this information to other proxy agents that located on different nodes, which enable the location proxy agent on each node to have global view of the whole system.

In experiment one, two communication scheme simulations, broadcast and location proxy are implemented in the experiment. The simulation is executed on a single cluster node, ASER1. To reduce the communication delay, all agents, including the system agents of JADE, are executed on the same JVM container. To measure the performance we utilize a *starter* agent which initiates the boid agent process and measures time. The running times for different number of agents is recorded using java's System.currentTimeMillis() method and the unit is milliseconds. Both simulations will be executed for ten times. The reported results are time average over 10 simulation runs of 1000 cycle each. The time reported does not include the overhead of starting and finishing agents, only the time consumed between boid start fly in the 2D space and end the fly after 1000 cycles. Experimental results are summarized in Figure 3.

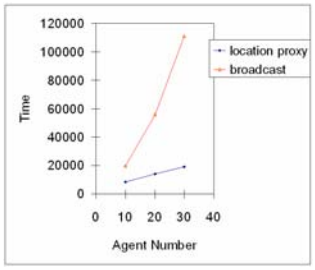


Figure 3: The performance of different communication schemes

As shown in Figure 3, the broadcast scheme requires more executing time than location proxy scheme. The broadcast scheme executing time increases very fast as the agent number increased. In broadcast scheme, each boid agent has to broadcast its current position to other agents in the virtual space at every step and collects the information broadcast by other agents to find out it neighbor agents. The communication complexity is O((n-1)!) at each time step. This scheme is not an efficient solution when a large amount of boid agents are simulated. In the location proxy scheme, all boid agents only report their new position to the location proxy agent when they move to a new place. The boid agent also inquires the location proxy agent for its nearby boid agent mates instead of searching by itself. In this communication scheme, boid agents do not communicate with each other. This scheme will largely reduce the communication cost within the agent group. The communication complexity is O(2n).

5.4 Experiment 2: Performance of the distributed MSFC implementation

The second experiment is to illustrate the performance enhancement by comparing the time cost for executing the same MSF MAB implementation on a three nodes cluster machine and a single processor machine. To reduce the impact of the JADE platform computation requirement, in both simulations, the JADE main container runs on head node of the cluster which is not count in the simulation nodes. In the three nodes cluster distributed model, the boid agents are equally distributed on three nodes and each node has one location proxy agent for collecting agent position. The location proxy on each node will exchange boid agents' position at every step. The architecture of the distributed model is shown in Figure 4.

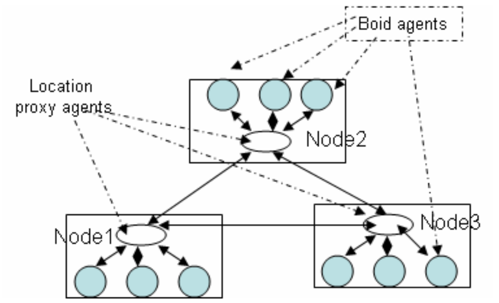


Figure 4: The architectures of multiple nodes implementation

Different numbers of boid agents are tested on both simulation and the boid agent's execute time for finishing 1000 circle is recorded. Because the distributed model requires three processes to simulate the document clustering, the time is the average time consummation for all agents running on different CPUs after 1000 cycles. The experiment results are shown in Figure 5. Aser 1-2-3 curve line in the chart indicates the time consummation of the clustering simulation executed on the three nodes cluster machine. Aser curve line in the chart indicates the time consummation of the clustering simulation executed on the single processor machine. Although the experiment shows three nodes simulation didn't cut the total time to one third of the total time of the simulation on single processor machine because of the overhead for agent update status with agent located on the same node or other node, still, the results show that the time saving of the distributed algorithm is well-suited scenarios scales.

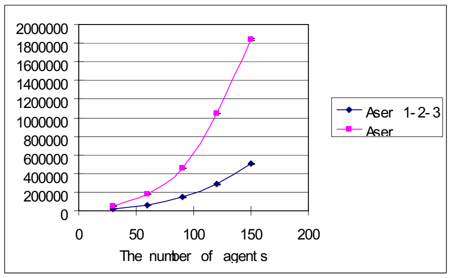


Figure 5: The executing time for 3 node cluster machine and one single processor machine

6. Conclusion

In this study, we proposed a new multiple species flocking (MSF) model and a distributed multi-agent MSF approach for document clustering. In this approach, each document in the dataset is represented by a bird agent. Each agent follows four simple local steering rules to move in the virtual space. Agents following these simple local rules emerge complex global behaviors of the whole flock and eventually the agents that carrying document belong to same class will gradually merge together to form a flock. All agents are evenly deployed on different nodes in a distributed computing environment for load balance. On each node, a location proxy agent is introduced for maintaining the agents' location and synchronizing the status between nodes in the cluster machine.

The advantage of the MSF clustering algorithm is the heuristic principle of the flock's searching mechanism. This heuristic searching mechanism helps bird agents quickly form a flock and reactive to the change of any individual document. Since the bird agent in the algorithm continues fly in the virtual space and join the flock it belongs to, new clustering results can be quickly re-generated when information stream is continually feed into the system.

7. References

[1] M. R. Anderberg, Cluster Analysis for Applications, Academic Press, Inc., New York, NY, 1973

[2] X. Cui, P. Palathingal, T. E. Potok, Document Clustering using Particle Swarm Optimization, IEEE Swarm Intelligence Symposium 2005, Pasadena, California, 2005, pp. 185-191

[3] X. Cui, T. E. Potok, A Flocking Based Algorithm for Document Clustering Analysis, Journal of System Architecture, Special issue on Nature Inspired Applied Systems, 2006. (To appear).

[4] B. Everitt, Cluster Analysis. 2nd Edition. Halsted Press, New York, 1980.

[5] G. Folino, G. Spezzano, "SPARROW: A Spatial Clustering Algorithm using Swarm Intelligence", Applied Informatics 2003 (AIA2003), Innsbruck, Austria, 2003 pp. 50-55.

[6] J. Handl and B. Meyer, Improved ant-based clustering and sorting in document retrieval interface, Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature, volume 2439 of LNCS, Springer-Verlag, Berlin, Germany, 2002, pp. 913–923.

[7] J. A. Hartigan, Clustering Algorithms. John Wiley and Sons, Inc., New York, NY, 1975.

[8] K. Jain, M. N.Murty, and P. J. Flynn, Data Clustering: A Review, ACM Computing Survey, Vol. 31, No. 3, 1999, pp. 264-323.

[9] G. Jones, A. Robertson, C. Santimetvirul, and P. Willett, Non-hierarchic document clustering using a genetic algorithm. Information Research, 1(1) (1995).

[10] V. D. Merwe and A. P. Engelbrecht, Data clustering using particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC2003), Canbella, Australia, 2003, pp. 215-220.

[11] V. Moere, Information Flocking: Time-Varying Data Visualization using Boid Behaviors, Proceedings of the Eighth International Conference on Information Visualization, 2004, pp. 409-414.

[12] M.F. Porter, An Algorithm for Suffix Stripping. Program, 14 no. 3, 1980, pp. 130-137.

[13] C. Reynolds, Flocks, Herds, and Schools: a distributed behavioral model, Computer graphics, 21(4), July 1987, pp. 25-34.

[14] S. Z. Selim, and M. A. Ismail, K-means type algorithms: A generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. Mach. Intell. 6, 1984, pp. 81–87.

[15] M. Steinbach, G. Karypis, V. Kumar, A Comparison of Document Clustering Techniques, Text Mining Workshop, KDD, 2000.

[16] J. Vesanto and E. Alhoniemi, Clustering of the Self−Organizing Map, IEEE Transactions on Neural Networks, Volume 11, Number 3, 2000, pp. 586−600.

[17] B. Wu and Z. Shi, A clustering algorithm based on swarm intelligence, Info-tech and Info-net Proceedings. ICII 2001, vol. 3, 2001, PP. 58-66.