A Bio-inspired Clustering Approach for Dynamic Document Distributed Analysis

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

Abstract

Document clustering is a fundamental operation used in unsupervised document organization, automatic topic extraction and information retrieval. But most clustering technologies are limited in their application on the static document collection. Intelligence analysts are currently overwhelmed with tremendous amount of text information streams generated everyday. There is a lack of comprehensive tool that can real-time analyze the dynamic changed information streams. In this paper, we propose a bio-inspired clustering model, the Multiple Species Flocking clustering model (MSFC), and present a distributed multi-agent MSFC approach for clustering dynamic updated text information streams. The decentralized architectures and communication schemes of the MSFC multi-agent distributed implementation for load balance and status information synchronization are also discussed in this article.

Keywords: Clustering, Bio-inspired, Flocking, Distributed, Load Balance

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 the 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 [12]. Document clustering is a fundamental operation used in unsupervised document organization, automatic topic extraction and information retrieval. Research in document clustering analysis mainly focuses on how to quickly and accurately cluster static document collection, while research on clustering the dynamic text information stream is limited. However, currently there is an increasing demand for clustering analysis of dynamic documents to meet the challenge of analyzing the extensive amount of text information stream generated everyday.

New algorithms based on biological models, such as ant colonies, bird flocks, and bee swarms, 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 following simple rules. Compared to traditional algorithms, the bio-inspired algorithms are usually flexible, robust, decentralized and self-organized. These characters make the bio-inspired algorithms potentially suitable for solving dynamic problems, such as dynamic changed document collection clustering. In 1991, Deneubourg [8] proposed one of the first clustering solutions inspired by ant colonies. Lumer and Faieta [13] extended Deneubourg's model to make it more suitable for data clustering. In recent years, more bio-inspired clustering approaches [3;4;7;21;23;24] were proposed for clustering data or document collection. However, these approaches still lack the ability to cluster dynamic changed document collection. In this paper, we propose a new bio-inspired clustering model, the Multiple Species Flocking clustering model (MSFC), and present a distributed multi-agent MSFC approach for clustering a dynamically updated text information stream. Unlike other partition

clustering algorithms such as K-means, the MSFC based algorithm does not require initial partitional seeds or cluster centers. This algorithm can continually refine the clustering results and quickly react to the individual data change. This character makes the MSFC algorithm capable of clustering a dynamically changed document information stream.

The remainder of this paper is organized as follows: Section 2 describes related work in the traditional and bio-inspired document clustering area. Section 3 provides preliminaries of document representation and document similarity computing in the clustering algorithms. Section 4 provides a general overview of the basic flocking model. The MSFC model is proposed and a single processor MSFC model implementation for clustering a static document collection is described in section 4. In section 5, a multi-agent scheme for distributed dynamic document clustering is presented. Section 6 provides detailed experimental setup and results to compare 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 7.

2. Related Work

There are two major clustering techniques: partitioning and hierarchical [12]. 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 [20]. The best-known partitioning algorithm is the K-means algorithm and its variants [19]. This algorithm is simple, straightforward and based on the firm foundation of analysis of variances. One drawback of the K-means algorithm is its clustering result is sensitive to the selection of the initial cluster centroids and may converge to 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. Some traditional partitional clustering algorithm variants, such as Fuzzy c-means (FCM) , are better than the K-means algorithm at avoiding local minima, but FCM can still converge to local minima [12]. The prior knowledge of the probable number of clusters and the fuzzy membership function is still required for implementing FCM.

To deal with the limitations that exist in the traditional partition clustering methods, inspired from biological collective behaviors, a number of computer scientists have proposed several approaches to solve the clustering problem, such as genetic algorithm (GA) [3], Particle Swarm Optimization (PSO) [7], Ant clustering [13] and Self-Organizing Maps (SOM) [23]. Among these clustering algorithms, the Ant clustering algorithm is a partitioning algorithm that does not require a prior knowledge of the probable number of clusters or the initial partition. Wu [24] and Handl [10;11] 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 the K-means algorithm. However, in the Ant clustering algorithm, the clustered data objects do not have mobility by 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 carrying 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 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. The experiment in [6] indicates the Flocking clustering algorithm is more efficient than the Ant clustering algorithm, because each document object in the collection is projected as an agent moving in a virtual space, and each agent's movement activity is heuristic as opposed to the random activity as in the Ant clustering algorithm. It does not require a prior knowledge of the number of clusters in the datasets as well. The major challenge of the Flocking clustering algorithm in [6] is how to distribute the algorithm’s computational load in a distributed environment. In the following section, we will explain how the MFSC model generated from bird flocks and how MSFC Multi-agent distributed implementation is applied to document clustering applications.

3. Preliminaries

3.1 Document Representation

In clustering algorithms, the dataset to be clustered can be 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 representing the object. The text document objects can then be represented using the Vector Space Model (VSM) [18]. In VSM, the content of a document is formalized as a dot in a multi-dimensional space and represented by a vector x, such as $x=\{w_1, w_2, \ldots..w_n\}$, where $w_i(i = 1,2,\ldots,n)$ is the term weight of the term t_i in one document. The term weight value w_i represents the significance of this term in a document. To calculate the term weight, the occurrence frequency of the term within a document and in the entire set of documents needs to be considered. The most widely used weighting scheme combines the term frequency with inverse document frequency (TF-IDF) [17]. The weight of term i in document j is given in Equation 1:

$$w_{ji} = tf_{ji} * idf_{ji} = tf_{ji} * \log_2(n / df_{ji}) \quad (1)$$

where tf_{ji} is the occurrence number of term i in the document j; df_{ji} indicates the term frequency in the document collections; and n is the total number of documents in the collection. Before translating the document collection into TF-IDF VSM, the very common words (e.g. function words: "a", "the", "in", "to"; pronouns: "I", "he", "she", "it") are completely stripped out and different forms of a word are reduced to one canonical form by using Porter's algorithm [14].

3.2 The Similarity Metric

The similarity between two documents needs to be measured in clustering analysis. Over the years, two prominent ways have been proposed to compute the similarity between documents x_p and x_j. The first method is based on Minkowski distances, given by:

$$D_n(x_p, x_j) = \left(\sum_{k=1}^{d_x} | w_{k,p} - w_{k,j} |^n \right)^{1/n} \quad (2)$$

where x_p and x_j are two document vectors; d_x denotes the dimension number of the vector space; $w_{k,p}$ and $m_{k,j}$ stands for the documents x_p and x_j's weight values in dimension k.

The other commonly used similarity measurement in document clustering is the cosine correlation measurement, given by:

$$\cos(x_p, x_j) = \frac{x_p x_j}{| x_p || x_j |} \quad (3)$$

where $x_p x_j$ denotes the dot-product of the two document vectors and |.| indicates the length of the vector.

Both similarity metrics have been widely used in the text document clustering literature. In the present proposed algorithm, we chose the Euclidean distance (Minkowski distances where n=2) as the similarity metric. In order to manipulate equivalent threshold distances, considering that the distance ranges will vary according to the dimension number, this algorithm uses the normalized Euclidean distance as the similarity metric of two documents, x_p and x_j, in the vector space. Equation 4 represents the distance measurement formula:

$$d(x_p, x_j) = \sqrt{\sum_{k=1}^{d_x} (w_{k,p} - w_{k,j})^2} \quad (4)$$

where x_p and x_j are two document vectors; d_x denotes the dimension number of the vector space; $w_{k,p}$ and $m_{k,j}$ stands for the documents x_p and x_j's weight values in the dimension k.

4. The Flocking Based Clustering Algorithm

4.1 The Flocking Model

A flock can be simply described as a group of individuals clustered together, moving with a common velocity. In nature, there are numerous examples of this sort of behavior including flocks of birds, herds of land animals, swarms of insects and schools of fish. The Flocking model is a bio-inspired computational model simulating the animation of a flock of entities, called "boids" [16]. In this model, each boid makes its movement decisions on its own according to a small number of simple rules that react to its neighboring members in the flock and the environment it senses. The Flocking model is one of the first collective behavior models that have many popular applications, such as animation [2], robotic control [5], time varying data visualization [15;22] and spatial cluster retrieval [9]. The basic Flocking model, first proposed by Craig Reynolds [16], consists of three simple steering rules that need to be executed at each instance over time, which includes: (1) Separation: steering to avoid collision with other boids nearby; (2) Alignment: steering toward the average heading and matching the velocity of its neighbor flock mates; (3) Cohesion: steering to the average position of the neighbor flock mates. These simple local rules of each boid generate complex global behaviors of the entire flock.

4.2 The Multiple Species Flocking Clustering Model

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

The following four mathematical equations illustrate these four action rule of each boid in the MSFC model:

Alignment Rule: $$d(P_x, P_b) \le d_1 \cap (P_x, P_b) \ge d_2 \Rightarrow \vec{v}_{ar} = \frac{1}{n}\sum_{x}^{n} \vec{v}_x \quad (5)$$

Separation Rule: $$d(P_x, P_b) \le d_2 \Rightarrow \vec{v}_{sr} = \sum_{x}^{n} \frac{\overline{\vec{v}_x + \vec{v}_b}}{d(P_x, P_b)} \quad (6)$$

Cohesion Rule: $$d(P_x, P_b) \le d_1 \cap (P_x, P_b) \ge d_2 \Rightarrow \vec{v}_{cr} = \sum_{x}^{n} (\overrightarrow{P_x - P_b}) \quad (7)$$

Feature Similarity Rule: $$v_{ds} = \sum_{x}^{n} S(B, X) * d(P_x, P_b) \quad (8)$$

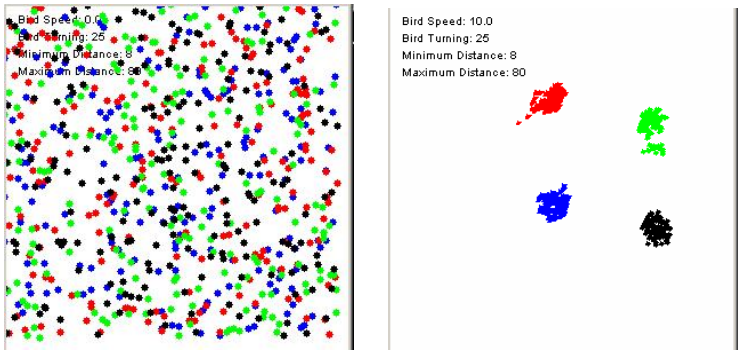
where $\boldsymbol{v_{ar}}$, $\boldsymbol{v_{sr}}$, $\boldsymbol{v_{cr}}$ and $\boldsymbol{v_{ds}}$ are velocities driven by the four action rules, $d(P_x,P_b)$ is the distance between boid B and its neighbor X, n is the total number of the boid B's local neighbors, $\boldsymbol{v_b}$ and $\boldsymbol{v_x}$ is the velocity of boid B and X**,** $\boldsymbol{d_1}$ and $\boldsymbol{d_2}$ are pre-defined distance values and $d_1 \succ d_2$, $\overrightarrow{P_x - P_b}$ calculates a directional vector point, ***S(B,X)*** is the similarity value between the features of boid B and X**.**

To achieve comprehensive flocking behavior, actions of all four rules are weighted and summed to give a net velocity vector demanded for the active flocking boid:

$$v = w_{sr}.v_{sr} + w_{ar}.v_{ar} + w_{cr}.v_{cr} + w_{ds}.v_{ds} \quad (9)$$

where v is the boid's velocity in the virtual space and $w_{sr}, w_{ar}, w_{cr}, w_{ds}, w_{dd}$ are pre-defined weight values.

Based on the MSFC model, we implemented a 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. In the simulation, 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 other boids within a limited range and move in the simulation environment by following the four action rules of the MSFC 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

4.3 The MSFC Algorithm for Document Clustering

Inspired by the bird's ability of maintaining a flock as well as separating different species or colony flocks, the Multiple Species Flocking Clustering (MSFC) algorithm uses a simple and heuristic way to cluster document datasets. In the MSFC 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. Following the simple rules in the MSFC model, each boid determines its movement on its own in the virtual space. Similar to a bird in the real world, the boids that share similar document vector features (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. In this algorithm, the behavior (velocity) of each boid is only influenced by its nearby boids. The four MSFC action rules react to this influence and generate the boid's new movement velocity. Although this influence on each boid is local, the impact on the entire boid group is global. After several iterations, these simple local rules followed by each boid result in generating a complex global behavior of the entire document flock, and eventually a document clustering result emerges.

The MSFC algorithm is a partitioning algorithm and does not require a prior knowledge of the number of clusters in the datasets. It generates a cluster of a given set of data through projecting the high-dimensional data items on a 2D grid for easy retrieval and visualization of the clustering result. In [6], we evaluated the efficiency of the MSFC algorithm with document collection that includes 100 news articles collected from the Internet. The news article collection has been categorized by human experts and manually clustered into 12 categories. For comparing

purpose, the Ant clustering algorithm and the K-means clustering algorithm are applied to the same real document collection dataset, respectively. The K-means algorithm requires prior knowledge about how many cluster are expected, which means the K-means algorithm needs to know the news article category number. Each algorithm will have 300 fixed iterations to refine the cluster results. The average results from ten separate experiments are listed in table 1. The results indicate that the flocking algorithm achieves better results when compared to the K-means and the Ant clustering algorithm for document clustering. The results also shows that the K-means clustering algorithm result is better than the Ant clustering algorithm result, which is different from the experiment results shown in [11;24]. This is due to the Ant clustering algorithm requiring more iterations for result refining. If given enough iterations, the Ant clustering algorithm can generate better results than the K-mean clustering algorithm.

Table 1: The performance results of K-means, Ant clustering and MSFC Algorithms

Algorithms	Average cluster result number	Average F-measure value
MSFC	10.083	0.8058
K-means	(12)	0.6684
Ant	1	0.1623

5. Multi-Agent Scheme for Distributed Dynamic Document Clustering

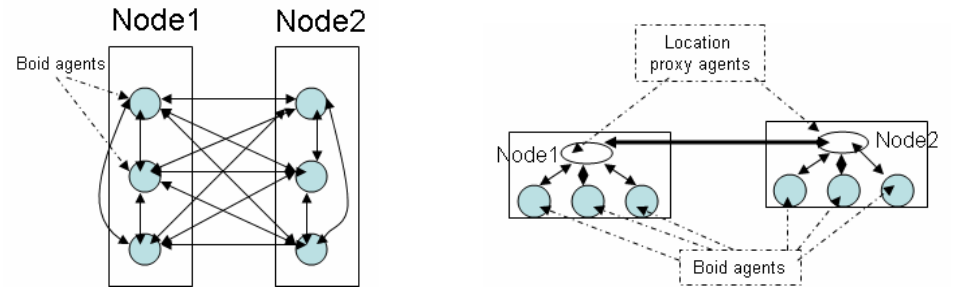
The continual movement of each boid in the MSFC algorithm makes the algorithm quickly react to the individual data change. This character indicates that the MSFC algorithm is suitable for clustering dynamic changed text stream. Inevitably, using a single processor machine to cluster the dynamic text stream requires a large amount of memory and faster execution CPU. Since the decentralized character of the MSFC model, using Multi-Agent techniques to develop a distributed MSFC clustering approach can increase the clustering speed of the algorithm.

In the MSFC approach, the document parse, similarity measure and boid moving velocity calculation are the most computational consumption parts. The distributed implementation will 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 need to be examined carefully when designing a distributed solution. The first issue is the load balance. It is important to maintain load balancing among processing nodes to make sure each node have approximately same workload. The second issue is the environment state synchronization. It is very important for a distributed implementation to develop a synchronization algorithm, which is capable of maintaining causality. The third issue is to reduce the communication between nodes, including the communication overhead of the environment state synchronization and the control message exchange between nodes. Based on these requirements, a distributed multi-agent based (MAB) implementation of the MSFC algorithm for clustering analysis of the text information stream was developed. In MAB, each boid is modeled and implemented in terms of agents, which makes each boid pro-active, adaptive and communicable. The MAB implementation support distributed load balancing among processing nodes in a very natural way. Since each boid agent is implemented 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 load balance. Since each agent can be added, removed or moved to other machine without interrupting other agent's running, the system will be scalability and pro-activity to the work load change.

One major concern in designing this distributed MAB MSFC system is how to ensure all the agents in the system be synchronized at any time when they need interact or exchange data. 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

communicate with other agents to collect enough information, does an exhaustive search to find out which agents are located within its range, and then calculates the force that it is pushed to travel based on its 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 easiest communication scheme to implement 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. In this scheme, each agent has a global view of the entire system status. However, each broadcast will use a lot of bandwidth and make the network bandwidth in a computer cluster become a bottleneck of the system as the agent number increases. Another scheme is the location proxy. As shown in Figure 2(b), there is a location proxy agent on each 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 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 a global view of the whole system.



(a) Broadcast (b) Location Proxy

Figure 2: The architectures of different communication schemes

6. Experiments and Results

6.1 Multi-Agent Platform

The distributed MSFC algorithm is implemented on a Java Agent DEvelopment Framework (JADE) agent platform. JADE is a software framework fully implemented in the Java programming 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 requirement is a Java virtual machine (JVM). Each JVM is a basic container of agents that provide a complete runtime environment for agents and allow several agents to concurrently execute on the same container, JVM. In principle, JADE allows multiple JADE containers run on the same host and agents can be deployed on different containers. However, our experiments indicate that the communication between agents located in different JADE containers is much slower than the communication between agents located in the same JADE container. To reduce the communication delay, all agents, including the system agents of JADE, are executed on the same container.

6.2 Datasets

The document dataset used in this report is derived from the TREC-5, TREC-6, and TREC-7 collections and represented as a set of vectors. As indicated in the previous session, the heuristic nature of MSFC enables this algorithm to continually refine the clustering results and quickly react to the change of the document contents. This character makes the algorithm suitable

for cluster analyzing dynamic changed document information. In this report, the performance of MSFC on clustering dynamic updated document collections is measured. To simulate the dynamic updated document collection, the document vector of each agent is periodically updated with a new document vector and the old document vector is considered as expired. For easily comparing the performance of different scenario, in the experiments, each agent's document feature will be updated for ten times during the entire life of the system execution. In each experiment, the system will run 1000 iterations and the average document update gap is 100 iterations.

6.3 Experiment setup

In the MSFC distributed implementation, each boid is implemented as a JADE agent. Each agent has the ability to calculate its moving velocity based on the four action rules as discussed in previous sessions. Each agent carries a feature vector representing a document vector. The environment used in the experiment consists of a continuous 2D plane, in which each boid is placed randomly on a grid within a 4000×4000 squire unit area. All experiments were carried out on an experiment Linux 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/). This software can real-time graphically display all cluster nodes' processor and network usage.

6.4 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 a virtual space. Two communication scheme simulations, broadcast and location proxy are implemented in the experiment. The simulation is executed on a single cluster node, ASER1. To measure the performance, we utilize a *starter* agent which initiates the boid agent process and measures the consumption time. The running time for different numbers of agents is recorded using java's System.currentTimeMillis() method and the unit is millisecond. Both simulations are executed for ten times. The reported results are average time 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 when the boids started flying in the 2D space and when the boids finished flying after 1000 cycles. Experimental results are summarized in Figure 3.

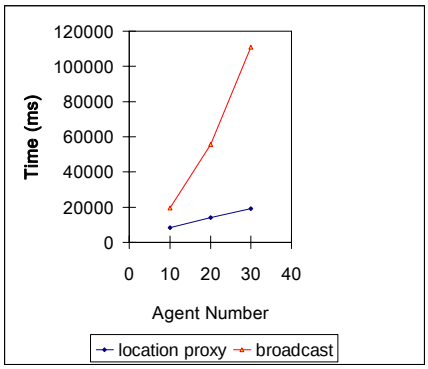


Figure 3: The performance of different communication schemes

As expected, the broadcast model has the worse performance and the running time increases very fast as the agent number increases. In this model, each boid agent has to broadcast its current position to other agents in the virtual space at every step and collects the information

broadcasted by other agents to find out its neighbor agents. The communication complexity is O((n-1)!) at each iteration. As such, the broadcast model is not an efficient solution if a large amount of boid agents are simulated. In the location proxy scheme, a location proxy agent is built in the simulation. All boid agents will report their new position to the location proxy agent once 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 within the agent group. The communication complexity is O(2n). However, the workload of the location proxy agent is largely increased because calculating each agent's nearby neighbors has to be completed by the location proxy agent instead of each boid agent. In a distributed environment, this means all boid agents have to wait for the location proxy agent to finish the calculation to find out their nearby neighbors.

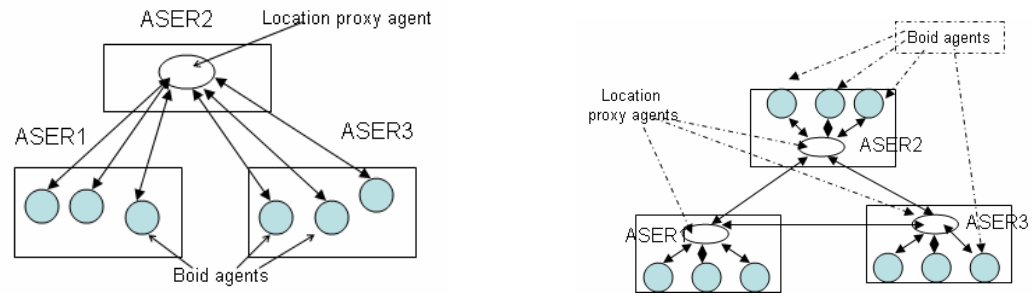
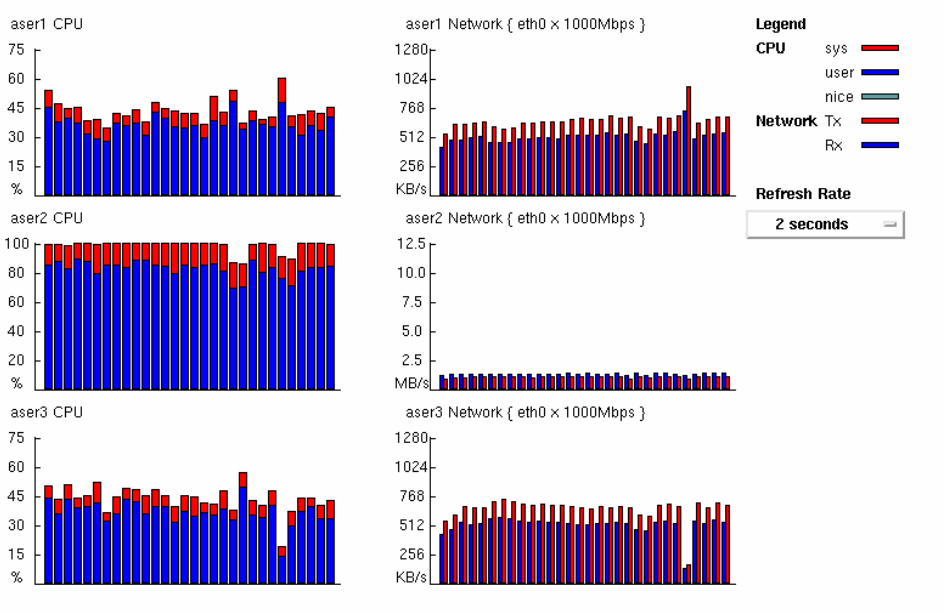


Figure 4: Location proxy agent deployment mechanisms

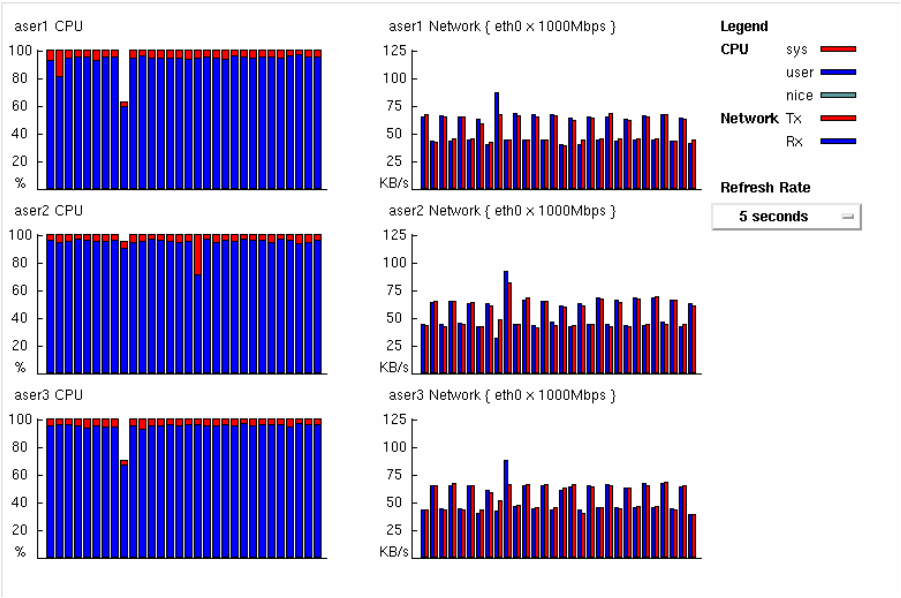
6.5 Experiment 2: Location Proxy Agent Deployment Mechanisms.

Experiment 2 illustrates the impact of different location proxy agent deployment on the network bandwidth and load balance in a distributed environment. In experiment 2, three cluster nodes, named as ASER1, ASER2, and ASER3, are involved to simulate different agent deployment mechanisms in distributed MSFC document clustering. Two kinds of location proxy agent deployment mechanisms are tested. In test 1, thirty boid agents are equally deployed on two cluster nodes, ASER1 and ASER3. One location proxy agent is executed on the cluster node ASER2. All boid agents report their location to the location proxy agent on ASER2 and inquire about their neighbor mates' status. Figure 4(a) shows the architecture of the central location proxy agent simulation. In test 2, the deployment of location proxy agents and boid agents follows the architecture described in Figure 4(b). In this architecture, all three cluster nodes are used for running the boid agents. Simultaneously, each cluster node has one location proxy agent in charge of the boid agents' location. The boid agents on each node report its new position to the location proxy agent in its home node. The location proxy agents on each node exchange the information with each other to make sure all location proxy agents have the same global view of the simulation environment.

The graphical chats of the CPU and network bandwidth usages of the two tests monitored by the LCM software are shown in Figure 5. As shown in the chart of Figure 5(a), the CPU usage of ASER2 is 100% because of the heavy workload of the location proxy agent. The CPU usage of ASER1 and ASER3 only reach 35% to 50% because most of the time the boid agents are waiting for the location proxy agent on node ASER2 to finish the calculation. The difference in CPU usage of three nodes indicates the unbalanced workload in the distributed environment. As shown in Figure 5(b), since the boid agents are equally distributed on three cluster nodes and each node has one location proxy agent, the CPU usage of each nodes are 100% , which indicates the workload of three nodes is balanced.



(a) One location proxy agent deployed on ASER2 node



(b) Three location proxy agents deployed on three nodes

Figure 5: The CPU and Network usage chats for different location proxy agent deployments

6.6 Experiment 3: Performance of the distributed MSFC implementation

The third experiment is to illustrate the performance enhancement by comparing the running time of executing the distributed MSFC implementation on a three-node 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 the head node of the cluster, which does not count as a simulation node. In the distributed model, the boid agents are equally distributed on three nodes and each node has one proxy server to collect agent's position and the proxy server on each node will exchange agents' position information at every step. The architecture of the distributed model is same as shown in Figure 4(b). Different numbers of boid

agents are tested on both simulations and the boid agent's execute time to finish 1000 circle is recorded. Because the distributed model requires three processes to simulate the document clustering, the time is the average consummation time for all agents running on different CPUs after 1000 cycles. The experiment results are shown in Figure 6. The "Three nodes" curve line in Figure 6 indicates the time consumption of the document clustering simulation executed on the three nodes cluster machine. The "One node" curve line indicates the time consummation of the document clustering simulation executed on one single node machine.

As shown in Figure 6, when the total boid agent number is below 100, the three nodes simulation didn't cut the total running time to one third of the total time of the simulation on a single processor machine because of the overhead for agent status updating. However, the running time on the single node machine increases faster than that on the three node machine. Once the total boid agent number is above 120, the time required for running on the single node machine is more than three times that on the three node machine. One possible reason is each node has limited memory (512M), when more than 100 agents running on the same node, depending on the documents that these agents represent, the memory requirement for the simulation may larger than the actual memory of the computer node, which causes the computer system to use virtual memory (hard disk space) and the time requirement to finish the simulation is largely increased. Our next step research will focus on 1) improving the boid agent's document feature representing method to reduce the memory requirement, 2) testing the performance the MSFC approach on a large number of cluster nodes and documents, and 3) building self-adaptive boid agents that can mobile between nodes to automatically balance the workload of each cluster node.

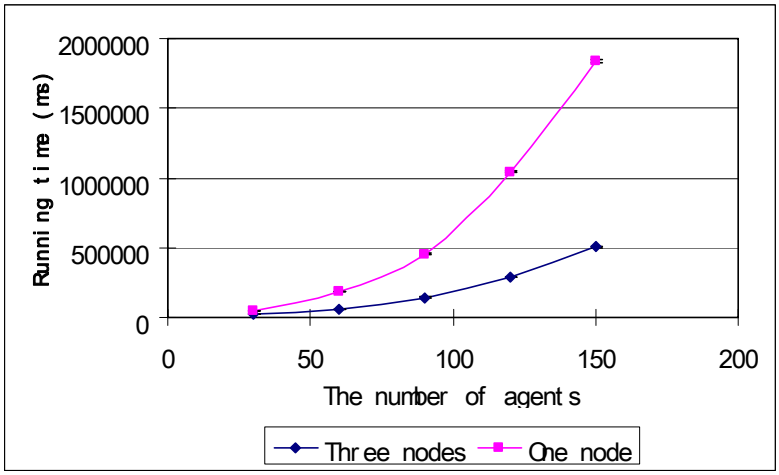


Figure 6: The running time for 3 node cluster machine and one single processor machine

7. Conclusion

In this study, we proposed a bio-inspired clustering model MFSC and presented a distributed multi-agent MSFC approach for dynamic document clustering analyzing. In this approach, each document in the dataset is represented by a boid agent. Each agent follows four simple local rules to move in the virtual space. Agents following these simple local rules emerge a complex global behavior of the entire flock and eventually the agents that carrying document belong to the same class will gradually merge together to form a flock. The advantage of the MSFC clustering algorithm is its heuristic principle of the flock's searching mechanism. This heuristic searching mechanism helps boid agents to quickly form a flock and react to the change of any individual document. Since the boid agent in the algorithm continues to fly in the virtual space and join the flock it belongs to, new results can be quickly re-generated when an information stream is

continually fed into the system. This approach is implemented on a JADE agent platform and tested on a three nodes cluster machine. Each JADE agent represents one boid agent in this approach. All agents are evenly deployed on different nodes in a distributed computing environment. Since each boid agent need to find out the location of other agents located in its nearby area, the agents' location information need to be exchanged at each iteration. To reduce the communication load of the location information exchange within boid agents, on each node, a location proxy agent is introduced to maintain the agents' location and synchronizing the status between nodes in the cluster machine. Our experimental results showed that the clustering process running on the three nodes is much faster than the process on a single node.

The MSFC algorithm can be used for clustering dynamical changed text information stream. It can also generate better results than the K-means clustering algorithm and another bio-inspired clustering algorithm, the Ant clustering algorithm, for clustering static document collection. However, the MSFC implementation in our experiment requires much more computational time and resources than the K-means implementation. In this paper, we presented a multi-agent distributed scheme to implement MFSC algorithm on distributed cluster machine to reduce the computational load of each single node. This multi-agent implementation can largely increase the MSFC clustering performance but still need long time when it is used to cluster large amount of text information stream. One reason is each boid agent need to be represented by one JADE agent and the JADE agent that we used in the experiment is a "heavy" agent, which consumes a large amount of computational resources when a great number of JADE agents are running simultaneously. Future work will consider the "light" agent implementation and use one "light" agent to represent all boid agents deployed on each node. That will largely reduce the computational resources required at each node and make the clustering of a large amount of document collections possible.

Acknowledgment

Oak Ridge National Laboratory is managed by UT-Battelle LLC for the U.S. Department of Energy under contract number DE-AC05_00OR22725.

References

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

[2] M. Anderson, E. McDaniel, and S. Chenney, "Constrained animation of flocks," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* San Diego, CA, USA: Assoc. for Comput. Machinery, 2003, pp. 286-297.

[3] A. Casillas, M. T. De Gonzalez Lena, and R. Martinez, "Document clustering into an unknown number of clusters using a genetic algorithm," in *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, 2807 ed Ceske Budejovice, Czech Republic: Springer Verlag, Heidelberg, D-69121, Germany, 2003, pp. 43-49.

[4] L. Chen, X. Xu, Y. Chen, and P. He, "A Novel Ant Clustering Algorithm Based on Cellular Automata," in *Proceedings - IEEE/WIC/ACM International Conference on Intelligent Agent Technology. IAT 2004* Beijing, China: IEEE Computer Society, Los Alamitos, CA 90720-1314, United States, 2004, pp. 148-154.

[5] B. Crowther, "Flocking of autonomous unmanned air vehicles," *Aeronautical Journal*, vol. 107, no. 1068 SPEC, pp. 99-109, 2003.

[6] X. Cui, J. Gao, and T. E. Potok, "A Flocking Based Algorithm for Document Clustering Analysis," *Journal of System Architecture*, no. Special issue on Nature Inspired Applied Systems July2006.

[7] X. Cui and T. E. Potok, "Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm," *Journal of Computer Sciences*, vol. Special Issue, pp. 27-33, 2005.

[8] J. L. Deneubourg, S. Goss, N. SendovaFranks, C. Detrain, and L. Chretien, "The dynamics of collective sorting robot-like ants and ant-like robots," Cambridge, MA, USA: MIT Press, 1991, pp. 356-363.

[9] G. Folino and G. Spezzano, "Sparrow: A Spatial Clustering Algorithm using Swarm Intelligence," in *IASTED International Multi-Conference on Applied Informatics*, 21 ed Innsbruck, Austria: Int. Assoc. of Science and Technology for Development, Calgery - Alberta, T3B OM6, Canada, 2003, pp. 50-55.

[10] J. Handl, J. Knowles, and M. Dorigo, "On the performance of ant-based clustering," in *Design and application of hybrid intelligent systems* Amsterdam, The Netherlands: IOS Press, 2003, pp. 204-213.

[11] J. Handl, J. Knowles, and M. Dorigo, "Ant-based clustering and topographic mapping," *Artificial Life*, vol. 12, no. 1, pp. 35-61, 2006.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, Sept.1999.

[13] E. D. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in *From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior* Brighton, UK: MIT Press, 1994, pp. 501-508.

[14] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130-137, July1980.

[15] G. Proctor and C. Winter, "Information flocking: data visualisation in virtual worlds using emergent behaviours," in *Virtual Worlds. First International Conference, VW'98. Proceedings* Paris, France: Springer-Verlag, 1998, pp. 168-176.

[16] C. W. Reynolds, "FLOCKS, HERDS, AND SCHOOLS: A DISTRIBUTED BEHAVIORAL MODEL," *Computer Graphics (ACM)*, vol. 21, no. 4, pp. 25-34, 1987.

[17] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513-523, 1988.

[18] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Cornell Univ., Ithaca, NY, USA, Copyright 1975, IEE CU-CSD-74-218, July1974.

[19] S. Z. Selim and M. A. Ismail, "K-MEANS-TYPE ALGORITHMS: A GENERALIZED CONVERGENCE THEOREM AND CHARACTERIZATION OF LOCAL OPTIMALITY," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 1, pp. 81-87, 1984.

[20] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," KDD workshop in text mining, 2000.

[21] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *2003 Congress on Evolutionary Computation (IEEE Cat. No.03TH8674)*, Vol.1 ed Canberra, ACT, Australia: IEEE, 2003, pp. 215-220.

[22] A. Vande Moere, "Information flocking: Time-varying data visualization using boid behaviors," in *Proceedings of the International Conference on Information Visualization*, 8 ed London, United Kingdom: Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855-1331, United States, 2004, pp. 409-414.

[23] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 586-600, May2000.

[24] B. Wu and Z. Shi, "A clustering algorithm based on swarm intelligence," in *2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479)*, vol.3 ed Beijing, China: IEEE, 2001, pp. 58-66.