A P2P-based Flocking Algorithm for Distributed Clustering using Small World Structure

Gianluigi Folino, Agostino Forestiero and Giandomenico Spezzano
CNR-Institute for High Performance Computing and Networking (ICAR)
Via P. Bucci 41c, I-87036 - Rende (CS), Italy

Abstract: Clustering has become an increasingly important task in modern application domains such as electronic commerce, multimedia, surveillance using sensor networks as well as many others. In many of these areas, the data are originally collected at different sites and their transmission to a central site is almost impossible. This requires to develop novel distributed clustering algorithms to handle the difficult problems posed from the dynamic topology changes of the network, impracticality of global communications and global synchronization and the frequent failure and recovery of resources. In this paper, we propose a biologically-inspired algorithm for clustering distributed data in a peer-to-peer network with a small world topology. The method proposed is based on a local flocking algorithm that uses a decentralized approach to discover clusters by a density-based approach and the execution, among the peers, of an iterative self-labeling strategy to generate global labels with which identify the clusters of all peers. We have measured the goodness of our flocking search strategy on performance in terms of accuracy and scalability. Furthermore, we evaluated the impact of small world topology in terms of reduction of iterations and messages exchanged to merge clusters.

Key–Words: Distributed Data Mining, Clustering, Bio-inspired algorithms, Small world paradigm, P2P networks

1 Introduction

Clustering algorithms have been applied to a wide range of problems, including exploratory data analysis, data mining, image segmentation and information retrieval. In all of these disciplines the common problem is that of grouping similar objects according to their distance, connectivity, or their relative density in space [5]. Traditional clustering methods hardly can be applied in the case of distributed datasets. Today's large-scale data sets are usually logically and physically distributed, requiring a distributed approach to mining and the development of approximate local algorithms [2].

Recently, many data mining algorithms based on biological models have been developed, like for instance [6], in order to solve the clustering problem. These paradigms are characterized by the interaction of a large number of simple agents that sense and change their environment locally. Ants' colonies, flocks of birds, termites, swarms of bees etc. are agent-based insect models that exhibit a collective intelligent behavior (*swarm intelligence*) [1] that may be used to define new distributed clustering algorithms. In these models, intelligent behavior frequently arises through indirect communication between the agents using the principle of **stigmergy**, taken from the insect societies. According to this principle an agent deposits something in the environment this makes no direct contribution to the task being undertaken but it is used to influence the subsequent behavior that is task related.

In this paper, we present P-SPARROW a multi-agent distributed clustering algorithm implemented in a small world P2P network which combines the stochastic search of an adaptive flocking with a density-based clustering method and an iterative self-labeling strategy to generate global labels with which identify the clusters of all peers.

P-SPARROW clusterizes data independently on different peers by a decentralized algorithm based on flocking hunting agents that execute in parallel a smart exploratory strategy to discover *local* models of the clusters represented by data points in which the cardinality of the points of the neighborhood exceeds a fixed threshold. Little by little that representative points are discovered they act as attractors for all the others. The entire flock then moves towards these representative points to explore neighboring regions that probably contain other representative points. Agents have different features (color and speed) to adapt their behavior according to their performance. For example, an agent poorly-performing speed-up in order to leave an empty or uninteresting part of the data space in order to find a more interesting area more quickly.

As soon as the representative points are discovered they are sent to the neighboring nodes to complete the clustering by a merge procedure. Clusters are merged using a global relaxation process in which nodes exchange cluster labels with neighboring peers until a fixed point (i.e. all nodes detect no change in the labels) is reached.

P-SPARROW has a number of nice properties. Its underlying topology exploits the potentialities of small world networks that have a high clustering coefficient and a low characteristic path length and this make the algorithm fault tolerant and permits a faster convergence. It works in an incremental way and this make it adapt to perform *approximate* clustering and its use as an *anytime* algorithm; it is completely decentralized, as each peer acts independently from each other. Furthermore, each peer communicates only with its immediate neighbors and the communication is asynchronous. Locality and asynchronism implies that the algorithm is scalable to very large networks.

We have implemented P-SPARROW using Java agents and the Jxta Protocol, for programming the communication and the synchronization among the P2P nodes. The remainder of this paper is organized as follows. Section 2 introduces P-SPARROW, presents the principles of the density-based algorithms and describes its architecture. Section 3 discusses the obtained results and the impact of the small world topology and Section 4 draws some conclusions.

2 P-SPARROW

In this section we present P-SPARROW a novel algorithm built upon the work [4] which uses the concepts of a flock of birds that move together in a complex manner using simple local rules, to cluster distributed homogeneous spatial data in P2P systems. Since P-SPARROW utilizes the principles of the conventional density-based clustering algorithms, the density-based method is described first, then the P-SPARROW algorithm is explained and finally the overall distributed architecture is illustrated.

2.1 Density-based clustering

Density-based clustering methods try to find clusters on the basis of the density of points in regions. Dense regions that are reachable from each other are merged to formed clusters.

DBSCAN [3] is one the most popular density based methods and it is based on the idea that all the points of a data set can be regrouped into two classes: *clusters* and *noise*. Clusters are defined as a set of dense connected regions with a given radius (*Eps*) and containing at least a minimum number (*MinPts*) of points. The two parameters, Eps and MinPts, must be specified by the user and allow to control the density of the cluster that must be retrieved. The algorithm defines two different kinds of points in a cluster: *core points* and *non-core points*.

A core point is a point with at least MinPts number of points in an Eps-neighborhood of the point. The non-core points in turn are either *border points* if they are not core points but they are *density-reachable* from another core point or *noise points* if they are not core points and are not density-reachable from other points. To find the clusters in a data set, DBSCAN starts from an arbitrary point and retrieves all points that are density-reachable from that point.

A point p is density reachable from a point q, if the two points are connected by a chain of points such that each point has a minimal number of data points, including the next point in the chain, within a fixed radius. If the point is a core point, then the procedure yields a cluster. If the point is on the border, then DBSCAN goes on to the next point in the database and the point is assigned to the noise. DBSCAN builds clusters in sequence (that is, one at a time), in the order in which they are encountered during space traversal. The retrieval of the density of a cluster is performed by successive spatial queries. Such queries are supported efficiently by spatial access methods such as R*-trees.

2.2 The P-SPARROW clustering algorithm

As in DBSCAN, P-SPARROW finds cluster performing region-queries on core points but it replaces the exhaustive search of the core points with a stochastic multi-agent search that discovers in parallel the points. P-SPARROW is constituted of two phases: a local phase for the **discovery** of the core points on each peer and a **merge** phase that concerns a global relaxation process in which nodes exchange cluster labels with nearest neighbors until a fixed point (i.e. all nodes detect no change in the labels) is reached.

Data are homogeneous and partitioned among the peers. Each peer implements the flocking algorithm, described in figure 1, using a fixed number of agents that initially occupy a randomly generated position in the space. Each agent moves testing the neighborhood of each object (data point) it visits in order to verify if the point can be identified as a *core point*. Then, P-SPARROW uses a flocking algorithm with an exploring behavior in which individual members (agents) search some goals, whose positions are not known *a priori*, in parallel and signal the presence or the lack of significant patterns into the data to other flock members, by changing color.

The entire flock then modes towards the agents

```
for i=1 . . . MaxIterations
    foreach agent (yellow, green)
        if (not visited (current_point))
            density = compute_local_density();
            mycolor= color_agent(density);
        endif
    end foreach
    foreach agent (yellow, green)
        dir= compute_dir();
    end foreach
    foreach agent (all)
    switch (mycolor){
        case yellow, green: move(dir, speed(mycolor)); break;
        case white: stop ();generate_new_agent();break;
        case red: stop (); merge(); if (new_red()) clone_agent(); break; }
    end foreach
        if ((bag_out.dimension()> threshold)or(i%IterMigr==0)) send_bag();
        if (bag_in_full()) notify_changes();
end for
```

Figure 1: The pseudo-code of P-SPARROW executed on every peer.

(*attractors*) that have discovered interesting regions to help them, avoiding the uninteresting areas that are instead marked as obstacles. The color is assigned to the agents by a function associated to the data analyzed during the exploration, according to the DBSCAN density-based rules and with the same parameters: *MinPts*, the minimum number of points to form a cluster and *Eps*, the radius of the circle containing these points. In practice, the agent computes the local density (density) in a circular neighborhood (with a radius determined by its limited sight, i.e. Eps) and then it chooses the color in accordance to the simple rules of figure 2.

$$\begin{array}{ll} density > MinPts & \Rightarrow mycolor = red\ (speed = 0) \\ \frac{MinPts}{4} < density \leq MinPts & \Rightarrow mycolor = green\ (speed = 1) \\ 0 < density \leq \frac{MinPts}{4} & \Rightarrow mycolor = yellow\ (speed = 2) \\ density = 0 & \Rightarrow mycolor = white\ (speed = 0) \end{array}$$

Figure 2: The rules for computing color and speed.

So *red*, reveals a high density of interesting patterns in the data, *green*, a medium one, *yellow*, a low one and white, indicates a total absence of patterns. The color is used as a communication mechanism among flock members to indicate them the roadmap to follow.

The main idea behind our approach is to take advantage of the colored agent in order to explore more accurately the most interesting regions (signaled by the red agents) and avoid the ones without clusters (signaled by the white agents). Red and white agents stop moving in order to signal these regions to the others, while green and yellow ones fly to find clusters. Green agents will move more slowly than yellow agents in order to explore more carefully zones with a higher density of points. The variable speed introduces an adaptive behavior in the algorithm. In fact, agents adapt their movement and change their behavior (speed) on the basis of their previous experience represented from the red and white agents. Anyway, each flying agent computes its heading by taking the weighted average of alignment, separation and cohesion (as illustrated in figure 3.

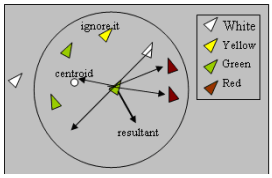


Figure 3: Computing the direction of a green agent.

Green and yellow agents compute their movement observing the positions of all the agents that are at most at some fixed distance (*dist_max*) from them and applying the rules of Reynolds' [7] with the following modifications: *alignment* and *cohesion* do not consider yellow agents, since they move in a not very attractive zone; *cohesion* is the resultant of the heading towards the average position of the green flockmates (centroid), of the attraction towards red agents, and of the repulsion by white agents; a *separation* distance is maintained from all the agents, whatever their color is.

New red agents executes the merge procedure; i.e., a temporary label will be given to these agents and to all the points of their neighborhood, if they are not already labeled. Otherwise the minimum of all the labels will be assigned to all the core points in this neighborhood, in order to make them belong to the same cluster. In this way, on each peer the set of red agents (core points) determinates the local model of clustering.

Neighboring peers must be informed about the new core points or about the new labels in order to merge all the points belonging to the same cluster. To this end, red agents create clone agents and put them in an apposite bag and, when a fixed number of clone agents is achieved (i.e. a bag of agents has reached the desired dimension) or a certain number of iterations have been performed, each peer will send the bag containing the cloned red agents to the neighbor-

ing peers. Consequently, the agents received from the other peers will be put in another bag that will be used in the next iteration (or when it become full) for the merge phase. In practice, the new agents continuously update the labels as multiple clusters take shape concurrently. This continues until nothing changes, by which time all the clusters will have been labeled with the minimum initial label of all the sites containing the data. All the points having the same label form a cluster.

2.3 The software architecture

The software architecture of P-SPARROW on one of the nodes of the P2P network is described in figure 4. On each node, the **flock platform** manages the cellular space in which the agents move. Furthermore, it supplies the main procedures concerning the agents (move, remote move, create new agent, clone agents, etc..) using the underlying levels. Agents of different colors will be scheduled by means of the **agents scheduler**.

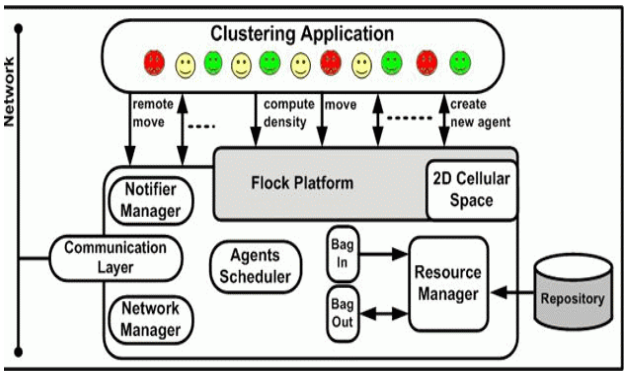


Figure 4: The software architecture of P-SPARROW on one of the nodes.

The **resource manager** (RM) execute efficiently range queries (i.e. compute density) in the dataset, accessing the repository, in order to choose the new color of the agents. The RM is also responsible of putting new agents received by the neighboring peers in the appropriate zone of the cellular space in order to start a new phase of merge. The arrival of a new bag of agents is signaled by the **notifier manager** that supplies also information about new events such as the fall of a peer, the convergence of the algorithm, etc... The **network manager** handles the send and the receive of the bags of agents on the basis of the topology of the system (see subsection 2.4 for more details about the topology used), using JXTA sockets.

2.4 Using Small World topology

In a first implementation, peers were arranged using a logical ring topology. However, using this topology the merge phase can waste many iterations before to merge all the clusters. This is due mainly to the characteristics of the ring topology, i.e. we have a high average hop count between two nodes. Furthermore, the fall of a node (event not infrequent in p2p networks) causes a dramatic increase in the hop count. On the other hand, if we used a completely connected topology, the network will be congested by the large number of messages exchanged.

An interesting alternative is using the small world topology [8] to describe the phenomenon that everyone in the world can be reached through a short chain of social acquaintances. We can characterize network topologies using two parameters: path length *CPL* (i.e. the length of the shortest path between each pair of nodes averaged over all possible pairs) and clustering coefficient *CC* (i.e. the ratio between the number of edges in the neighborhood of a node and the total number of possible edges averaged over all nodes). Small world topology, showed in figure 5 (b) have a high CC but a low CPL and that is a really useful property in p2p networks, as we need a few hops to reach a node, but the disconnection of a node does not change the behavior/performance of the system.

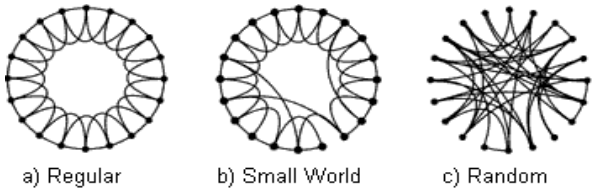
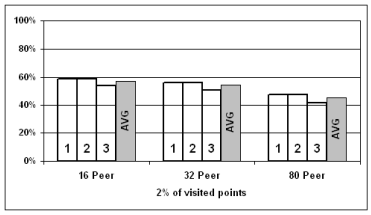


Figure 5: (a) Regular Lattice ($\beta = 0$) (b) Small World ($\beta = 0.1$) (c) Random ($\beta = 0.8$)

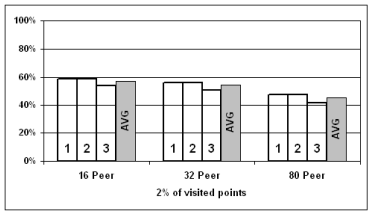
The topology of the network depends on the parameter β, if it tends to zero we will have a regular lattice topology, if it varies from 0.01 to 0.1 we will have a small world topology and higher values will bring to a random topology (figure 5 c). In subsection 3.2 we reported the improvements obtained using the small world topology.

3 Experimental Results

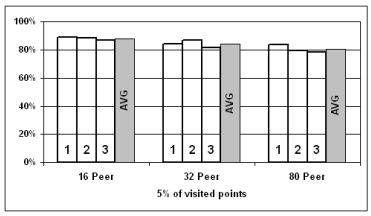
In this section, we want to analyze the goodness of our algorithm in the task of performing approximate clustering and we want to verify some interesting properties of our distributed system (i.e. accuracy, scalability, etc..). In our experiments, we used a real spatial dataset called Sequoia. Sequoia is a dataset com-



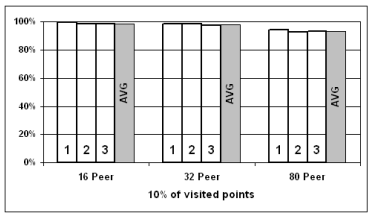
(a)



(b)



(c)



(d)

Figure 6: Number of points for cluster for Sequoia dataset (percentage in comparison to the total number of points for cluster) when P-SPARROW analyzes 1%, 2% 5% and 10% of total points, using 16, 32 and 80 peers.

posed by 62556 names of landmarks (and their coordinates), and was extracted from the US Geological Survey's Geographic Name Information System. The three main clusters in this dataset represent respectively the areas of S. Francisco, Sacramento and Los Angeles.

3.1 Accuracy and Scalability

We run our algorithm using 100 agents working until they explore the 1%, 2%, 5% and 10% of the entire data set, using 16, 32 and 80 peers. All the experiments were averaged over 30 trials. Our algorithm uses the same parameters as DBSCAN. Therefore, if we visited all the points of the dataset, we would obtain the same results as DBSCAN, as the merge phase is the same. Then, in our experiments we consider as 100% the cluster points found by DBSCAN (note DBSCAN visit all the points). We want to verify how we come close to this percentage visiting only a portion of the entire dataset and that must be effective for different number of peers involved in the computation. Note that the dominant operation in the computation is the execution of the range queries, performed each time a point is visited, while the time of the other operations is negligible. So, the fact of reducing the percentage of visited points considerably reduces the execution time.

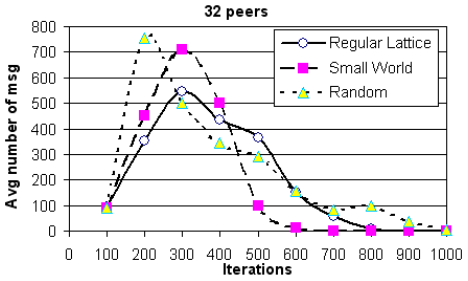
For a large number of peers, the density of points for cluster for peer necessarily decreases; so we have to choose a different value of the parameter MinPts to keep into account this aspect. In practice, we choose a value of MinPts inversely proportional to the number of peers (i.e. if we fix MinPts as 8 on 16 peers, we have to fix as 4 on 32 peers and so on). In figure 6, we show the experimental results concerning the accuracy and scalability of the algorithm by varying the number of peers for the Sequoia dataset. For instance, on 80 peers, visiting only the 5% of points, on average, we obtain an accuracy of 80% and visiting the 10% of data we reach 93% of accuracy. Furthermore, the scalability (i.e. the effect on the accuracy of increasing the number of peers and so reducing the number of data points for peer) is quite good. In fact, if look at the Sequoia dataset, for the 5% case, we obtained a reduction from 88% for 16 peers to 81% for 80 peers while for the 10% case, we have a little reduction from 99% to 94%. Visiting only 1% of the dataset we have low percentage of point found, however they are sufficient to have an approximate idea of shape of the clusters.

For instance, on 80 peers, visiting only the 5% of points, on average, we obtain an accuracy of about 80% for the Sequoia dataset. Furthermore, visiting the 10% of data we reach 93% of accuracy.

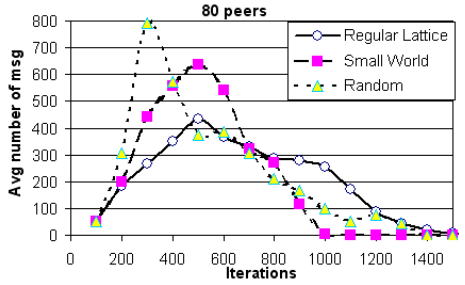
Moreover, the scalability (i.e. the effect on the accuracy of increasing the number of peers and so reducing the number of data points for peer) is quite good for the dataset. In fact, if look at the Sequoia dataset, for the 5% case, we obtained a reduction from 88% for 16 peers to 81% for 80 peers while for the 10% case, we have a little reduction from 99% to 94%. Visiting only 1% of the dataset we have low percentage of point found, however they are sufficient to have an approximate idea of shape of the clusters.

3.2 The impact of Small World topology

In order to test the effect of the small world topology, we run P-Sparrow with the same parameters re-



(a)



(b)

Figure 7: Avg number of messages exchanged for the three different topologies (regular lattice, small world and random) using 32 and 80 peers for Sequoia.

ported in the previous subsection using the three different topologies of figure 5 ($\beta = 0$, $\beta = 0.1$ and $\beta = 0.8$) with the real datset Sequoia. Note that we used two configurations respectively of 32 and 80 peers, as 16 peers are not sufficient to generate a small world topology.

In figures 7 (a) and (b), we reported the average number of messages exchanged for peer (peers exchange core points each 100 generation) for the three different topologies for the Sequoia dataset. Note that when the number of messages approach to 0 means that the algorithm does not discover new solutions (core points) and then it converged. The SW topology reach a higher peak in comparison with the regular (but less than the random), but then converges more quickly, probably because of the effect of the long-range links that accelerates the diffusion of the core points and then the process of clustering. In the case of the random topology, the low clustering coefficient disperses many core points and this slow down the convergence.

4 Conclusions

This paper describes P-SPARROW, a algorithm for distributed clustering of data in peer-to-peer environments combining a smart exploratory strategy based on a flock of birds with a density-based strategy to discover clusters of arbitrary shape and size in spatial data. The algorithm has been implemented in a peer-to-peer system and evaluated using a real word dataset. Experimental results show that P-SPARROW can be efficiently applied as a data reduction strategy to perform approximate clustering. Moreover, the algorithm scales well when the number of peer increases. Finally, the use of a small world topology helps the algorithm to merge clusters more quickly with a slightly higher number of messages exchanged and permits a better fault tolerance because of high clustering coefficient characteristic of this topology.

References:

[1] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. Swarm intelligence: From natural to artificial systems. *J. Artificial Societies and Social Simulation*, 4(1), 2001.

[2] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta. Distributed data mining in peer-to-peer networks. *IEEE Internet Computing*, 10(4):18–26, 2006.

[3] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[4] Gianluigi Folino and Giandomenico Spezzano. An adaptive flocking algorithm for spatial clustering. In *PPSN*, pages 924–933, 2002.

[5] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, September 2000.

[6] Nicolas Monmarché, M. Slimane, and Gilles Venturini. On improving clustering in numerical databases with artificial ants. In *ECAL '99: Proceedings of the 5th European Conference on Advances in Artificial Life*, pages 626–635, London, UK, 1999. Springer-Verlag.

[7] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM Press.

[8] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.