A decentralized gossip based approach for data clustering in peer-to-peer networks

Rasool Azimi a, Hedieh Sajedi b, *

a Young Researchers and Elite Club, Qazvin Branch, Islamic Azad University, Qazvin, Iran
b Department of Computer Science, School of Mathematics, Statistics and Computer Science, College of Science, University of Tehran, Tehran, Iran

HIGHLIGHTS

• A distributed data clustering method is proposed based on the gossip communications.
• The proposed approach is adaptive to dynamic condition of unstructured P2P networks.
• The peers perform clustering only through communications with their local neighbors.
• This method no need to use central servers and sync operations between the peers.
• The evaluation results verify the accuracy of the proposed method with low overhead.

ABSTRACT

In this paper, a novel distributed approach, named GDSOM-P2P, for clustering distributed data resources is proposed by combining, an improved version of Silhouette algorithm, the dynamic Self-Organizing Map (SOM) neural network, and VICINITY protocol as a generic overlay management framework based on self-organization. The proposed GDSOM-P2P is adapted to the dynamic conditions of these networks. In the proposed GDSOM-P2P algorithm, at first, each node extracts a number of important data through the SOM and Silhouette algorithms. Then each of the nodes chooses one of its neighbors with the help of the VICINITY algorithm, and exchanges their important data with their neighbors. By doing this, over a period, the nodes’ data will be distributed over the entire network and the nodes in the network access the summary data model of the whole data. Finally, each node aggregates its internal data with a summary model and then performs the final clustering to cluster its internal data correctly. Evaluation results over a real P2P environment verify the efficiency of proposed GDSOM-P2P. Furthermore, the proposed GDSOM-P2P is also compared with the existing well-established distributed data clustering techniques. The results show a significant accuracy improvement of the proposed method.

© 2018 Elsevier Inc. All rights reserved.

1. Introduction

Recent advances in computing and communication over wired and wireless networks have resulted in many dynamic pervasive distributed computing environments. These environments often come with different distributed sources of data. The analysis of large and distributed data sources has become an important tool in understanding complex environments in areas such as economics, business, science and engineering [29].

Effective data mining method in such environments naturally calls for proper utilization of these distributed resources. Unfortunately, most off-the-shelf data mining techniques are designed to work as a monolithic centralized application. They work only on data that can be accessed in its entirety. Application of these techniques for mining data distributed in a network requires downloading the relevant data to a centralized location before performing the data mining operations. However, data centralization may not be possible in all cases because of network or storage constraint. Moreover, in many privacy sensitive applications different multi-party datasets collected at different sites must be processed in a distributed fashion without collecting everything to a single central site. Thus, this centralized data mining approach does not work for many of the emerging distributed, ubiquitous, possibly privacy-sensitive data mining applications [7]. Distributed Data Mining is well-understood as a resource-intensive and time-consuming task which is devoted to extract patterns and regularities from huge amounts of distributed datasets [5].

Peer-to-peer (P2P) computing is emerging as a new distributed computing paradigm for novel applications that involves exchange
of information among peers with little centralized coordination. Analyzing data distributed in P2P networks requires peer-to-peer data mining algorithms that can mine the data without data centralization [13].

The main contribution of this work is to introduce a novel distributed data clustering algorithm that with the help of non-interconnected peers and the dissemination of a series of data samples at the network level and with the help of a weighted clustering approach, provides data mining results within a desired accuracy level, and shows good scalability with low communication overhead, without the need for the synchronization, search, and other complex processes.

The paper is organized as follows: Section 2 presents the literature survey on approaches related to the proposed method. Section 3 introduces our distributed clustering algorithm for unstructured P2P networks, called GDSOM-P2P. In Section 4, the proposed GDSOM-P2P algorithm is evaluated in both static and dynamic situations over a real P2P network. Section 5, concludes the paper.

2. Related works

So far, several methods have been proposed for clustering data in distributed environments.

In [10], a distributed data clustering algorithm for distributed document clustering is represented, which is able to classify documents throughout a distributed P2P network. In short, we call this algorithm DP2P K-Means. In this approach, a K-Means clustering algorithm [14,15] is used for practical classification of documents and a Probe/Echo mechanism for distribution of tasks throughout P2P networks and disseminating results to the cluster starter. Initiator peer in clustering operation estimates a primary set of centroids and sends these centroids together with a Probe message, to all its neighbors. Each peer receives a Probe message just once and sends it to all its neighbors, except for the neighbor from which, it has received the Probe message. Then, a K-Means clustering is applied in the locally available documents. When the peer receives the Echo message from one of its neighbors, it merges the results of its own clustering with the clustering results sent by the neighboring peer through the Echo message. This merger is accomplished by “Weighted Mean” of centroids, in which the weight of each centroid relates to the number of the documents assigned to that centroid, in each cluster. After each peer receives Probe and Echo messages from all its neighbors, it sends the results of the clustering, as centroids of clustering and their weights, through an Echo message, to the peer from which it has first received the Probe message. When each starter peer receives Probe and Echo messages from all its neighbors, it has perfect information on the current iteration of K-Means algorithm. Hence, the current iteration is completed and terminated. Typically, a complete K-Means clustering algorithm requires more than one iteration. Thus, based on the situation, the starter peer decides that, whether or not the currently achieved clustering should be improved through an additional iteration. In the next iteration, the starter peer sends centroids of the produced cluster and their weights after the last iteration to its neighbors through a Probe message, with the new name “Guess”, and the work continues. The results show the high speed processing of this algorithm in comparison to centralized processing, for such datasets, as text, and saving the transmission time, which by itself justifies the use of a distributed approach. One of the problems of this algorithm may be its need for global synchronization. In this way, because indexing all terms in the DHT is too expensive, only a small number of manually selected terms are indexed. Nevertheless, it requires extensive human interaction, and the network is not able to adapt itself to new documents and topics dynamically. A brief review of this approach is presented in Table 1.

In [6], the P2P K-Means algorithm is proposed for distributed data clustering in large-scale networks. Unlike [10], this algorithm just requires local synchronization. In this method, the sync operation of each of the nodes is accomplished locally through their immediate neighbors. The algorithm begins with a peer that randomly produces primary centroids and sends them to all its immediate neighbors, together with a termination threshold, which is invariable and defined by the user, and initiates the first iteration. When a peer receives the primary centroids and the termination threshold, it sends them to other its neighbors, and starts the first iteration. Finally, all peers, with those primary centroids and termination threshold, enter the first iteration of the algorithm. Afterward, in each iteration, each peer receives from its neighboring peers, centroids, and the number of clusters of each iteration. Then, using this information and iterating them with local data, this peer produces centroids for the next iteration. If the distance between new centroids and centroids of the previous iteration is more than the termination threshold, the next iteration will begin. Otherwise, the peer will enter the termination phase. This algorithm is communication efficient and robust to network or data changes. In this approach, the achieved accuracy is relatively high, but the communication cost is high. A brief review of this work is shown in Table 2.

In [8] proposed an improved version of P2P K-Means algorithm [6], named LSP2P algorithm. This algorithm is among the clustering methods based on partitioning, dedicated to P2P networks and the dynamic issues of these networks are considered. This algorithm does not use traditional data collecting methods based on aggregating the local models and perform the clustering operation in a central site. Instead, it does use, the local interactions between peers. The algorithm starts with a peer N0, and this peer randomly produces an initial set of centroids, transmits this set along with a termination threshold limit (a constant defined by the user) to all of its immediate neighbors, and begins the first iteration. Once a peer receives initial centroids and termination threshold, it therefore transmits them to its immediate neighbors and iteration starts. Eventually, all peers start the first iteration of the algorithm with the same initial centroids and termination threshold. In summary, the LSP2P K-Means algorithm is the frequent iterations of improved K-Means algorithm in every peer Nk. Each peer Nk receives, in each iteration, centroids and the number of clusters related to iteration l from neighboring peers. This peer, then produces centroids for the next iteration using this information and its integration with local data. If the distance between centroids of the new clusters and centroids of the previous iteration is greater than the threshold, the next iteration is started. Otherwise, that peer will reach the termination state. When peer Nk leaves the network (in fact, it is eliminated from the network), its immediate neighbors notice this because of monitoring. In this case, any of the Nk peers realize that their immediate neighbor Nk have been removed, removes peer Nk from the waiting list and gets rid of any information request message Nk from the request table. When a new peer of Nk is added to the network, the associated peers will discover it. Nk will need to synchronize itself with the immediate neighbors and then the algorithm continues as before. The experimental results of the accuracy study of clustering showed that the algorithm reached a very good accuracy at the end of each period. This algorithm is completely distributed data clustering approaches such that the central server is not used and the peers interact locally with their immediate neighbors for clustering in the network. Based on the results of the experiment, this algorithm manifested excellent scalability, yet the performance of the algorithm was not affected by network size. This algorithm considers the dynamics of P2P networks. In discussing the communication, it is shown that the communication complexity in general (total bytes are exchanged on the network during execution time of the
algorithm) increases linearly with the size of the networks. The main problem with this algorithm is that, it is not possible to guarantee the analytical accuracy of the algorithm and this motivated to propose USP2P K-Means algorithm [15]. A brief review of the LSP2P k-Means algorithm is presented in Table 3.

In [8] a completely distributed clustering algorithm called USP2P K-Means was also proposed considering the fact that the accuracy of LSP2P K-Means algorithm cannot be guaranteed. This algorithm is actually K-Means algorithm based on uniform sampling in P2P networks. In any iteration of the algorithm, S of peers are selected by uniform sampling and then used to update centroids. Synchronization is also performed, similar to LSP2P, for USP2P, except that synchronizing action is applied to USP2P only for a subset of the peers for each iteration. As mentioned earlier, unlike LSP2P, the accuracy of USP2P is guaranteed. However, the accuracy is guaranteed only if the data and network are not changed from the beginning to the end of the algorithm. The algorithm assumes that a peer act as a coordinator of iterations. Peer \( N_1 \) selects arbitrarily the initial set of points \( V_1 = \{v_{1j} : 1 \leq j \leq k \} \) for K-Means algorithm. This peer has an accuracy threshold limit of cluster \( c \), a probabilistic variable of ensuring cluster accuracy \( p \), and size of peers selected for sampling \( s \), and the maximum number of iterations \( M \) specified by the user. Assuming that the data was centralized in the network and \( V_1 \) was used as initial centroids, and \( V_1^{(s)} = \{v_{1j}^{(s)} : 1 \leq j \leq k \} \) indicated centroids results, the standard K-Means algorithm is at the beginning of the iteration \( I \geq 1 \) as \( V_1^{(s)} = V_1 \). At the start of iteration \( I \), \( N_1 \) estimates centroids in \( V_1^{(s)} \), including \( V_1 = \{v_{1j} : 1 \leq j \leq k \} \). Peer \( N_1 \) calculates \( V_{i+1} \) in iteration \( I \). \( N_1 \) selects \( s \) number of network peers in each iteration by uniform sampling. Suppose that \( I_1 \) is the set of peers \( s \) plus \( N_1 \), peer \( N_1 \) transmits the initial set of points \( V_i \) (in a point-to-point connection) for \( I_1 \) (selected peers by \( N_1 \) using uniform sampling). Then each peer of \( N_i \in I_1 \) returns the obtained results, i.e., centroids and their weights to peer \( N_1 \) through an execution of K-Means algorithm on its local data. Once \( N_1 \) receives centroids and their weights from each peer \( N_i \), it then calculates the \( V_{i+1} \) using a weighted average. Now, peer \( N_1 \) transmits the results \( V_{i+1} \) as initial set of points for the next iteration to the same peers (\( N_i \in I_1 \)). In fact, these peers of \( N_i \) are the result of new sampling of \( N_i \) from set \( I_1 \). Peer \( N_1 \) considers a data dependent random variable \( c_i \) in order to guarantee the accuracy of the cluster in iteration \( I \). Moreover, peer \( N_i \) considers an accuracy, ensuring threshold \( c \). In iteration \( I_1 \), the cluster accuracy \( c_i \) is below the accuracy-ensuring threshold \( c \), the algorithm terminates. Peer \( N_i \) also holds the pre-assumption number of iterations \( M \), and if the algorithm iteration reaches \( M \), algorithm terminates. This algorithm has the accuracy, ensuring and guarantees clustering accuracy by calculating the threshold level. However, the main challenge of the algorithm is that, unlike the LSP2P method, it lacks the dynamics of P2P networks and operates by a pre-assumption that data and network will not change during the implementation of the algorithm. Another challenge of this algorithm is the uniform distribution of data amongst the peers in a network environment, and thus it is not appropriate for some large-scale networks with non-uniform distribution. USP2P uses sampling to provide probabilistic guarantees. However, the probabilistic guarantees are based on the assumption that data is uniformly distributed among the peers and when data are not uniformly distributed, a large percentage of the peers have to be involved in the communication in order to achieve a good approximation. In addition, USP2P requires a coordinating peer, which gets easily overloaded, since it is responsible for exchanging centroids with a significant number of peers. A brief review of the USP2P K-Means is shown in Table 4.

In [18], a distributed density-based clustering algorithm, named GoSCAN is proposed. It is a weighted version of DBSCAN [11] over a network consisting of \( n \) nodes with parameters \( \mu \text{minPts} \) and \( \varepsilon \). In DBSCAN, a data item is marked as a core point if it has at least \( \mu \text{minPts} \) data items within its \( \varepsilon \) radius. In addition, two core points are within one cluster, if they are in \( \varepsilon \) range of each other, or are connected by a chain of core points, where each two consecutive core points have a maximum distance of \( \varepsilon \). A non-core data item located within \( \varepsilon \) distance from a core point, is in the same cluster as that core point, otherwise it is an outlier. The GoSCAN algorithm can be broken into two major tasks: 1. Identifying core points, and 2. Forming the actual clusters. These two tasks are performed in a parallel employing gossip-based communication. This method reaches to higher accuracy in a network with fewer peers compared to networks with large amount of peers. A brief review of the GoSCAN is shown in Table 5.

Ref. [9] proposes a parallel, scalable adaptation of the classical K-Means algorithm for cluster analysis. Indeed, as the authors recognize, while the straightforward parallel formulation of K-Means is well-suited for distributed-memory systems with reliable interconnection networks, such as massively-parallel processors and clusters of workstations, in large-scale geographically-distributed systems the straightforward parallel algorithm can be rendered useless by a single communication failure or high latency in

---

**Table 1**

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Brief review of DP2P K-means algorithm [10].</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td>An approach for distributed document clustering based on K-moons on P2P Network</td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td>- Self-organization; - Scalability</td>
</tr>
<tr>
<td><strong>Weaknesses</strong></td>
<td>- Scalability; - Need for global synchronization</td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Brief review of the P2P algorithm [15].</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td>Distributed data clustering on large-scale dynamic P2P Network</td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td>- No global synchronization required; - Being communication-efficient; - Being robust to network or data changes</td>
</tr>
<tr>
<td><strong>Weaknesses</strong></td>
<td>High cost of communication</td>
</tr>
</tbody>
</table>

**Table 3**

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Brief review of the LSP2P K-means algorithm [6].</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td>A distributed K-means clustering algorithms based on local synchronization of P2P Network</td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td>- No global synchronization required; - Considering the dynamics of P2P network; - Able to deal with topology changes and loss of data</td>
</tr>
<tr>
<td><strong>Weaknesses</strong></td>
<td>- Clustering accuracy not be guaranteed; - Not robust to outliers</td>
</tr>
</tbody>
</table>
communication paths. As a consequence, the lack of scalable and fault tolerant global communication and synchronization methods in large-scale systems has hindered the adoption of K-Means in the context of applications over large networks such as Wireless Sensor Networks, Peer-To-Peer Systems and Mobile Ad-Hoc Networks. In line with this breaking evidence, the paper proposes a fully distributed K-Means algorithm, called Epidemic K-Means, which does not require global communication and it is intrinsically fault tolerant. The proposed distributed K-Means algorithm provides a clustering solution which is capable of approximating the solution of an ideal centralized algorithm over the aggregated data as closely as desired. A comparative performance analysis is carried out by the authors against the state-of-the-art sampling methods and shows that the Epidemic K-Means overcomes the limitations of the sampling-based approaches for skewed cluster distributions. The results of this experimental analysis confirm that the proposed algorithm is highly accurate and fault-tolerant under unreliable network conditions (e.g., message loss and node failures), and that it is suitable for asynchronous networks of very large and extremely scale. However, the performance of Epidemic K-Means decreases under losing messages and failure peers in the asynchronous networks. A brief review of the Epidemic K-Means algorithm is presented in Table 6.

In [17] a fully decentralized data clustering algorithm, named GDCluster, is proposed. In this method, peers continuously cooperate through decentralized gossip-based communication to maintain summarized view of the dataset. The proposed algorithm enabled nodes to gradually build a summarized view on the global dataset, and execute weighted clustering algorithms to build the clustering models. In general, because of the need for the pervasive communications, the lack of scalability and the lack compatibility with dynamic systems, the GDCluster lack the desired features to be implemented in large scale P2P environments. However, due to the high computational overhead, the efficiency of GDCluster decreases in large-scale networks as internal data of peers increase. In addition, applying the classical K-means method may lead to wrong clustering results. The performance of K-means clustering is dependent on the initial centroids, which are randomly selected in the first phase of the algorithm, and it is often trapped in local minima due to its hill climbing approach (see Table 7).

Among the latest work on distributed clustering, In [4] a novel canonical correlation analysis (CCA) framework equipped with L1-norm and L2-norm regularization terms is proposed to cluster the sensor data based on their information content. This method is a combination of Block coordinate descent (BCD) and the alternating direction method of multipliers (ADMM) framework to derive a centralized algorithm tackling the novel regularized CCA framework. Further, splitting of the regularized CCA into localized minimization subtasks across sensors enables distributed clustering of heterogeneous data based on their information content. Numerical tests demonstrate that the novel framework can achieve a higher probability of correct clustering than existing alternatives. However, the computation cost of ADMM method is high due to complicated mathematics for every gradient.

3. The proposed GDSOM-P2P algorithm

The proposed GDSOM-P2P algorithm is a hybrid approach using by combining, the improved Silhouette algorithm, the dynamic self-organizing map (SOM) approach [21], and VICINITY protocol [27] as a generic overlay management framework based on self-organization.

Consider a set \( P = \{ p_1, p_2, \ldots, p_n \} \) of \( n \) peers in the network. Each peer \( p_i \) stores an internal dataset \( D^{\text{int}}_{p_i} \) and shares the whole or part of it with other nodes. The internal data of the nodes can change over time. If we consider all the data of all nodes in the network as a general dataset, this set includes: \( D = \bigcup_{p_i \in P} D^{\text{int}}_{p_i} \).

The purpose of presenting the GDSOM-P2P algorithm is to cluster the entire dataset with a distributed approach, without the need to aggregate node data in a central server for performing...
traditional data mining methods. The proposed method is unnecessary to any local and global synchronization and has a lower communication cost than the previous pinhole methods.

3.1. Start of proposed GDSOM-P2P

At the beginning of the algorithm, each peer \( p_i \) should select a number of data to share with other nodes, through its internal data \( D_{p_i} \) using the improved Silhouette algorithm and the dynamic self-organizing map (DSOM) approach.

3.1.1. Improved Silhouette algorithm

The improved Silhouette approach [1] is used to determine the number of representative data in each peer. The modified silhouette algorithm is as follows:

1. Specify \( K_{\text{min}} \) and \( K_{\text{max}} \) values. Default values: \( K_{\text{min}} = 2 \) and \( K_{\text{max}} = \lceil \sqrt{n} \rceil \) where \( n \) is the number of input patterns.
2. For each iteration \( m \), calculate the initial centroids using the proposed initialization method and assign each of the transformed data \( Y \) to the nearest initial centroid to form initial clusters. Then calculate the mean of all data in each cluster as new centroids \( C_i = [c_{i1}, \ldots, c_{ik}] \).
3. Cluster the input data using any clustering technique for each iteration \( m \) \( K_{\text{min}} \leq m \leq K_{\text{max}} \).
4. Calculate \( S_m^n \) for each of the input data at each iteration \( m \), \( K_{\text{min}} \leq m \leq K_{\text{max}} \) by:

\[
S_m^n = \frac{|a(i) - b(i)|}{\max(a(i), b(i))}
\]

5. Where \( a(i) \) is the distance between the \( i \)th data \( (1 \leq i \leq n) \) and the nearest centroid \( c_{ij} \) \( (1 \leq j \leq K) \) at the \( K \)th iteration; \( b(i) \) is the minimum distance of the \( i \)th data from the other \( K - 1 \) centroids at the \( K \)th iteration. The proposed definition of \( a(i) \) and \( b(i) \) by Eq. (1) decreases the computational burden and speeds up the process as compared to their original definitions of the silhouette algorithm [22].
6. Include \( S_m^n \) for the \( i \)th data at the \( m \)th iteration in the \( S_m \) array.
7. Include the average value of \( S_m^n \) (for the \( m \)th iteration) in the \( m \)th cell of the array \( S_{\text{ave}} \).
8. Use Eq. (2) and select the row number with the highest \( S_{\text{ave}} \) as the estimated number of representatives.

\[
K_{\text{est}} = \text{ArgMax} \left[ S_{\text{ave}}, m \right] \quad \text{...(2)}
\]

3.1.2. DSOM approach

The DSOM algorithm is essentially a variation of the SOM algorithm where the time dependency has been removed. Regular learning function \( (3) \) and neighborhood function \( (4) \) have been respectively replaced by Eqs. (7) and (8):

\[
\Delta w_i = \varepsilon(t) \eta \left( i, s, v \right) \left( v - w_i \right)
\]

\[
\eta \left( i, s, v \right) = \frac{1}{\exp \left( \frac{-\left\| v - w_i \right\|^2}{2 \sigma_i^2} \right)}
\]

with \( \eta \left( i, s, v \right) \) being a neighborhood function of the form

\[
\eta \left( i, s, v \right) = e^{(-1) \left\| v - w_i \right\|^2} \quad \text{(4)}
\]

where \( s(t) \in \mathbb{R} \) is the learning rate and \( \sigma(t) \in \mathbb{R} \) is the width of the neighborhood defined as

\[
\sigma(t) = \sigma_i \left( \frac{\sigma_f}{\sigma_i} \right)^\eta \quad \text{with} \quad \varepsilon(t) = \frac{\varepsilon_f}{\varepsilon_i} \quad \text{and} \quad \eta \approx \varepsilon_i.
\]

Replacing functions (7) and (8) with functions (2) and (3) reflect two main ideas:

1. If a neuron is close enough to the data, there is no need for others to learn anything: the winner can represent the data.
2. If there is no neuron close enough to the data, any neuron learns the data according to its own distance to the data.

This draws several consequences on the notion of neighborhood that is now dynamic and leads to a qualitatively different self-organization that can be controlled using a free elasticity parameter.

DSOM is a neural map equipped with a structure (a hypercube or hexagonal lattice) and each neuron \( i \) is assigned a fixed position \( p_i \in \mathbb{R}^q \) where \( q \) is the dimension of the lattice (usually 1 or 2). The learning process is an iterative process where vectors \( a \in \Omega \) are sequentially presented to the map with respect to the probability density function \( f \). For each presented vector \( a \), a winner \( s \in N \) is determined according to Eq. (6).

Where a neural map is defined as the projection from a manifold \( \Omega \subset \mathbb{R}^q \) onto a set \( N \) of \( n \) neurons which is formally written as \( \Phi: \Omega \mapsto N \). Each neuron \( i \) is associated with a code word \( w_i \in \mathbb{R}_d \) all of which establish the set \( \{w_i\} \in N \) that is referred as the codebook. The mapping from \( \Omega \) to \( N \) is a closest-neighbor winner-take-all rule such that any vector \( v \in \Omega \) is mapped to a neuron \( i \) with the code \( w_i \) being closest to the actual presented stimulus vector \( v \).

\[
\phi: v \mapsto \text{arg min}_{i \in N} \left( \left\| v - w_i \right\| \right).
\]

The neuron \( w_i \) is called the winning element and the set \( C_i = \{x \in \Omega | \phi(x) = w_i \} \) is called the receptive field of the neuron \( i \). The geometry corresponds to a Voronoi diagram of the space with \( w_i \) as the center.

All codes \( w_i \) from the codebook \( W \) are shifted towards \( v \) according to:

\[
\Delta w_i = \varepsilon \left\| v - w_i \right\| \eta \left( i, s, v \right) \left\| v - w_i \right\|
\]

with \( \varepsilon \) being a constant learning rate and \( \eta \left( i, s, v \right) \) being a neighborhood function of the form:

\[
\eta \left( i, s, v \right) = \frac{1}{\exp \left( \frac{-\left\| v - w_i \right\|^2}{2 \sigma_i^2} \right)}
\]
Fig. 1. View of the whole dataset includes 3000 data (blue dots) and summarized view (yellow dots). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where \( \eta \) is the elasticity or plasticity parameter. If \( v = w_s \), then

\[
\eta \sigma(i, s, v) = 0.
\]

In this paper, DSOM approach is used to select representative data in each peer \( p_i \).

The process of selecting \( S_i \) representative data from each node \( p_i \) is as follows:

1. Run the improved Silhouette algorithm over the \( p_i \)’s internal data to determine the number of representative data for sharing with other nodes.
2. Run the DSOM algorithm over the internal data by considering \( S_i \) as input parameter.
3. Selection of the \( S_i \) samples from the \( p_i \)’s internal data that are peer-to-peer, closer to the \( S_i \) output data of the DSOM algorithm based on the Euclidean distance criterion. It should be noted that duplicate selective data is deleted.

By performing this process, the \( S_i \) data are returned as \( P \)’s representative data for sharing with other peers. Fig. 1 shows the complete flowchart of the process for selecting the \( S_i \) representative data in each peer \( p_i \).

Consider a dataset consisting of 3000 data and 15 clusters in a network consisting of 100 peers, where each peer \( p_i \) has 30 (\( D_{pi}^{int} = 30 \)) local data. After implementing the proposed method to determine the representative data in each of the nodes, 184 data were returned as representative data (shared data). Fig. 1 shows the total data and a summary sample derived from the implementation of the proposed algorithm.

In order to evaluate the quality of the representative data, we aggregate the representative data of all peers with each other as a summary view of total data. Then, to determine the quality of the summary view, we run the DSOM clustering algorithm twice. 1: Over the whole data and 2: Over the summarized view. As shown in Fig. 2, the clustering results were close to each other in both cases, and this represents the quality of the proposed method for determining representative data. It should be noted that here we assumed that the shared data of the peers are fully published on the network.

As shown in Fig. 2, the centroids obtained through the whole dataset are almost consistent with the centroids obtained from the summary view. This reflects the high quality of the summarized view provided by the proposed method.

3.2. Gossip-based interactions

In the process of the GDSOM-P2P algorithm, after the \( p_i \) node has selected its representative data, it needs to share these data with its neighboring nodes through the Gossip-based VICINITY protocol, with the goal that these representative data be shared throughout the network. In this step, each node \( P \), as an active peer, attempts to communicate with one of its neighbors \( Q \), as a passive peer, via the VICINITY communication protocol to share their representative data with each other.

3.2.1. VICINITY

Vicinity is an advanced gossip protocol that constructs an extra layer on top of an unstructured layer, that preserves the randomness in the network and keeps everyone connected in an evenly distributed manner.

Consider a connected network infrastructure, supporting routing between any two nodes. Each node maintains a dynamic list of neighbors, called its VICINITY view \( V_{vic} \), of fixed small length. The view length, \( \ell_{vic} \), is the same for all nodes. Knowledge regarding neighbors is stored and exchanged by means of node descriptors. A given node’s descriptor can only be created by that node exclusively. A node descriptor referring to peer \( P \) is a tuple containing the following three fields:

1. \( P \)’s contact information (i.e., network address and port)
2. A numeric age field
3. \( P \)’s application-specific profile.

Note that a VICINITY node descriptor is essentially a CYCLON node descriptor augmented by the node’s application-specific profile and a node’s profile determines its neighbors in the target structure. The goal is to organize all VICINITY views to approximate the target structure as closely as possible. To this end, nodes regularly exchange node descriptors to gradually evolve their views towards the target. When gossiping, nodes send each other a subset of their views, of fixed small length \( g_{vic} \), known as the gossip length. The gossip length is the same for all nodes.

Consider a selection function \( S(k, P, D) \), that, given the descriptor of peer \( P \) and a set \( D \) of peer descriptors, returns the set of \( k \)
descriptors (or all of them, if \(|D| < k\)) that best approximate \(P\)'s outgoing links in the target structure.

The selection is based on node profiles. We assume function \(S\) to be globally known by all nodes in the system. The selection function essentially defines the target structure.

Each peer \(P\) aims at eventually establishing links to the “best” \(\ell_{\text{vic}}\) peers, as defined by the outcome of \(S(\ell_{\text{vic}}, P, D_P^*)\), where \(D_P^*\) is the set of descriptors of all nodes in the network excluding \(P\).

Often, the selection function \(S\) is based on a globally defined peer proximity metric.

That is, \(S(k, P, D)\) sorts all descriptors in \(D\) with respect to their proximity to peer \(P\), and selects the \(k\) closest ones.

Typical proximity metrics include semantic similarity, ID-based sorting, domain name proximity, geographic- or latency based proximity, etc. Some applications may apply composite proximity metrics, combining two or more of the above. In certain cases, though, selecting appropriate neighbors involves more than a mere sorting based on some metric, typically when a peer’s significance as a neighbor depends not only on the peer’s proximity to a given node, but also on which other peers are being selected.

Assume that the selection function \(S\) exhibits some sort of transitivity, in the sense that if node \(P_2\) is a “good” selection for node \(P_1 (P_1 \rightarrow P_2)\), and \(P_3\) is a “good” selection for \(P_2 (P_2 \rightarrow P_3)\), then \(P_3\) tends to be a “good” selection for \(P_1\) too (\(P_1 \rightarrow P_3\)).

Generally, the “better” a selection node \(Q\) is for node \(P\), the more likely it is that \(Q\)’s “good” selections are also “good” for \(P\).

This transitivity is essentially a correlation property between nodes sharing common neighbors, embodying the principle “my friend’s friend is also my friend”. Surely, this correlation is fuzzy and generally hard to quantify.

It is more of a desired property rather than a hard requirement for our topology construction framework. The framework excels for networks exhibiting strong transitivity.

However, its efficiency degrades as the transitivity becomes weaker. In the extreme case that no correlation holds between nodes with common neighbors, related nodes eventually discover each other through random encounters, although this may take a long time.

3.2.2. VICINITY’s design rationale

According to the concept of protocol in the overlay networks, we are seeking a means to construct, for each node and with respect to the given selection function, the optimal view from all nodes currently in the system.

There are two sides to this construction:

First, based on the assumption of transitivity in the selection function \(S\), a peer should explore the nearby peers that its neighbors have found.

In other words, if \(P_2\) is in \(P_1\)’s VICINITY view, and \(P_3\) is in \(P_2\)’s view, it makes sense to check whether \(P_3\) would also be suitable as a neighbor of \(P_1\).

Exploiting the transitivity in \(S\) should then quickly lead to high-quality views.

The way a node tries to improve its VICINITY view resembles hill-climbing algorithms.

However, instead of trying to locate a single optimal node, here the objective is to optimize the selection of a whole set of nodes, namely the view.

In that respect, VICINITY can be thought of as a distributed, collaborative hill-climbing algorithm.

Second, it is important that all nodes are examined.

The problem with following transitivity alone is that a node will be eventually searching only in a single cluster of related peers, possibly missing out on other clusters of also related (but still unknown) peers, in a way similar to getting locked in a local maximum in hill-climbing algorithms.

\[ V_{\text{vic}} = V_{\text{vic}} - \{Q\}. \]
3. Merger the VICINITY and PEER SAMPLING SERVICE views in one: \( V_p = V_{vic} \cup V_{pss} \).
4. Add own descriptor with own profile and age 0 to the merged view: \( V_p = V_p \cup \{P\} \).
5. Strip down \( V_p \) to its \( g_{vic} \) best descriptors for \( Q \), by applying the selection function from \( Q \)’s perspective: \( V_p = S(g_{vic}, Q, V_p) \).
6. Send \( V_p \) to peer \( Q \).
7. Select 5 representative data to peer \( Q \) using proposed method introduced in Section 3.1.
8. Send representative data to peer \( Q \).
9. Similarly, receive \( VQ \) from peer \( Q \), containing a set of (up to) \( G_{vic} \) descriptors known by \( Q \), optimally selected for \( P \).
10. Receive \( S \) representative data from peer \( Q \).
11. Merge the VICINITY, PEER SAMPLING SERVICE, and received views in one: \( V = V_{vic} \cup V_{pss} \cup V_Q \).
12. Rebuild the VICINITY view by selecting the best \( \ell_{vic} \) neighbors from \( V \): \( V_{vic} = S(\ell_{vic}, P, V) \).
13. Merge the received \( S \) shared data from peer \( Q \) with the local data.
14. Summarize the local data if the \( P \)’s memory is full.

The receiving node \( Q \) executes all steps from 3 in a symmetric way. Note that when merging views (steps 3 and 11), in the presence of multiple descriptors of the same node only the one with the lowest age is kept. Each node essentially runs two threads.

An active one, which periodically wakes up and initiates communication to another peer, executing steps 1 through 14.

Another thread, a passive one, responds to the communication initiated by another peer, and executes steps 3 through 14.

The overall process of the activities of a network node \( P \) up to reach the final centroids is shown in Fig. 5. Moreover, the gossiping skeleton of proposed GDSOM-P2P based on VICINITY protocol is introduced in Fig. 6 and Table 8 lists the description of each hook of the skeleton, for the VICINITY protocol.

### 3.4. Summarization

Generally, each time the node’s memory is full, the nodes performs the summarizing operations over the local memory using the Cluster sampling method as follows:

1. Divide the local data into groups (clusters) using DSOM method.
2. Obtain a random sample of so many clusters from all possible clusters. (Clusters that more data is allocated to them).
3. Obtain a fraction of the total data on every sampling unit in each of the randomly selected clusters as summarized model.

### 3.5. Adapting to dynamic network conditions

To adapt the algorithm to the dynamics of the network, at the time of gossip-based interactions among peers, an AGE variable is considered for each shared data of peers. Thus, at the first round of gossip-based interactions, when the data is given, for the first time, as a representative of a peer to another peer, the receiving peer assigns the data an age variable. At each round of gossip-based operations, all peers increment one unit the age of their external data. Therefore, after a while, old external data are excluded from external data of peers by the end of their lifetime, and new data replace them. If the peer’s external data contain duplicate data with the same age, the data will be removed. If duplicate data have different ages, the younger one will be removed.
3.6. The final clustering results in each peer

In this section, we proposed a new method for determining the number of final clusters and the location of the final cluster centroids using the DSOM algorithm and the concept of the topological map in the dynamic self-organizing map neural network. We call this topological map-based clustering algorithm TMB-Cluster in brief.

Consider a dataset consisting of 2000 data and 15 clusters. After implementing the DSOM method over the input pattern, according to Fig. 7, we faced with several separated networks with interconnected data.

For a set of \( N \) data (vertices), we consider a \( N \times N \) weight matrix, where each matrix element \( n_{d_i, d_j} \) indicates the connection status between the two vertices \( d_i \) and \( d_j \), we have:

\[
\begin{align*}
    n_{d_i, d_j} = 1, \text{ status: connected} \\
    n_{d_i, d_j} = 0, \text{ status: disconnected}.
\end{align*}
\] (9)

If we consider the corresponding matrix as a network topology, the number of interconnected networks, or, in other words, the number of connected graphs, indicates the number of clusters. As shown in Fig. 7a, there are 15 separate networks or connected graphs which is shown with red edges. Each of connected graphs is exactly within a specific cluster.

In addition, in order to cluster the entire data, it is sufficient to calculate the mean of the vertices of each of the connected graphs separately that these obtained points are the final cluster centroids.
Finally, the entire data can be clustered by assigning each data to the nearest cluster centroid.

**Fig. 8** shows the cluster centers obtained through the proposed method. Also based on the results of clustering, each cluster is shown in a different color.

Whenever any peer \( p_i \) needs the final results of clustering, perform following process of proposed TMB-Cluster algorithm over its aggregated data \( D_{Agg}^p \):

1. Run the DSOM algorithm over the \( D_{Agg}^p \) by considering a high value default number as codebook in the dynamic self-organizing map neural network to form the topological structure (Map) of the data.
2. Determine the number of connected graphs, as the number of clusters.
3. Calculate the mean of the vertices of each of the connected graphs separately as the final cluster centroids.
4. Assign, the each of \( D_{Int}^p \) data to the nearest centroid.

**Fig. 8.** The clustering results of the proposed method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

By doing this process, corresponding peer cluster its local data \( D_{Int}^p \) based on the summarized view received from the network’s peers.

### 4. Evaluation results

In this section, we evaluate the performance of the proposed method to deal with the empty cluster problem; then we assess the proposed TMB-Cluster clustering algorithm and we evaluate the accuracy of the proposed GDSOM-P2P. To assess the efficiency of proposed algorithms, synthetic and real data were used. Each cluster in the synthetic datasets consists of a skewed set of data composed from two Gaussian distributions with different values of mean and standard deviation. In some synthetic datasets, there are 5% random noise data for challenging the clustering accuracy of algorithms. These datasets were already used in [18] and [17].

The real datasets used in the experiment are available online at Joensuu [23], uci [25] websites. More information regarding the data is presented in Fig. 9.

#### 4.1. Evaluation results of TMB-Cluster algorithm

This section evaluates the accuracy of the proposed TMB-Cluster. The AC value is calculated to compare the clustering accuracy of proposed TMB-Cluster with several state of the art and well-established clustering algorithm.

#### 4.1.1. Accuracy (AC) criterion

Suppose that \( C \) shows the actual centroids. By implementation of a clustering algorithm, \( K \) centroids of \( C^c = \{C^c_1, C^c_2, C^c_3, \ldots, C^c_K\} \) are calculated. According to Eq. (3), the AC criterion result is a number in the range \([0, 1]\). If this number is closer to one, indicates that calculated centroids are closer to the actual centroids. It means the high clustering accuracy and as a result, more accurate clustering is done [28].

\[
AC = \frac{\sum_{d \in D} \delta(C(d), \text{map}(C^c(d)))}{|D|} \tag{10}
\]
|D| is the total amount of network data. The map \(C^p(d)\) function is used for mapping a calculated cluster \(C^p\) to the actual cluster \(C\). The \(d(x, y)\) function is “1” if \(x = y\), otherwise, it returns the value of zero.

The comparative evaluation results of the proposed TMB-Cluster algorithm against with other state of the art and well-established clustering algorithm include the \(K\)-means with random init, \(K\)-medoids, \(K\)-means with Forgy init, \(K\)-means with Mac Queen init, \(K\)-means with Kaufman init, \(K\)-means with Refinement init, \(K\)-means with MDC init, Fuzzy C-means, \(K\)-Means*, Min–Max \(K\)-means algorithm and several improved versions of \(K\)-means algorithm, including Forgy, MacQueen, Kaufman \cite{20}, Refinement \cite{3}, MDC \cite{2}, \(K\)-Means* \cite{16} and Min–Max \(K\)-means algorithm \cite{24} over the Birch1, IRIS and Wine and New Thyroid datasets are shown in Figs. 10, 12–14, respectively.

Given that mentioned algorithms are not able to determine the number of clusters, to make a fair comparison, the number of clusters is assumed constant.

The clustering result of proposed TMB-Cluster over the Birch1 dataset is shown in Fig. 11.

The evaluation results show that the proposed TMB-Cluster improves the quality of the clustering in comparison with other clustering methods.

4.2. The evaluation results of proposed GDSOM-P2P

This section evaluates the performance of the proposed GDSOM-P2P algorithm in both static and dynamic network situations. The AC, Randl measures are used to assess the performance of the proposed GDSOM-P2P.

4.2.1. The Rand index (Randl)

Suppose “a” is the number of pairs of elements that are in the same real cluster. Also in the same computed cluster, while “b” is the number of pairs of elements that are in different real clusters and in different computed clusters. According to Eq. (11), Randl criterion \cite{12} identify the similarities between two clusters (as real and computed cluster) as follows:

\[
\text{Randl} = \frac{a + b}{\binom{n}{2}}.
\]
Fig. 12. Comparing the clustering results over IRIS dataset.

Fig. 13. Comparing the clustering results over Newthyroid dataset.

Fig. 14. Comparing the clustering results over Wine dataset.
4.2.2. The GDSOM-P2P Clustering accuracy evaluation procedure

The evaluation process of proposed distributed data clustering algorithm is as follows: first, the AC and RandI value for obtained centroids at each peer is calculated, and then the mean of AC results in the total network peers are considered as the evaluation results of distributed data clustering approaches.

4.2.3. The communication/storage overhead

The communication storage cost is measured in terms of average amount of data (in KB) transmitted/stored by each node, per gossip round. Which includes gossip packages containing the addresses of neighbor peers and the packages containing external data and in addition, the internal data in each peer [17].

4.3. Simulation results

Due to the highly decentralized and distributed nature of P2P networks, such as hundreds of peers, which typically join and leave continuously, it is prohibitively expensive to test a P2P algorithm by deploying it on real-world networks and evaluating the proposed algorithm performance. To be able to conduct research on evaluation of P2P protocols, simulation is often used as it can provide beneficial results without the expense of creating a real network.

To evaluate the approach proposed in this paper, we use PeerSim simulator [19] which is an open source discrete event simulator tool written in java. This simulation tool provides both cycle-driven and discrete-event simulation, and supports for structured and unstructured P2P models. PeerSim environment allows modeling hundreds of thousands of independent nodes on a normal desktop PC with just 1–2 GB of RAM.

For simulation, we created an unstructured peer topology and then run algorithm over static and dynamic network settings. In both cases, we evaluate the performance of proposed GDSOM-P2P when the internal data and number of peers varies. Table 9 shows parameter values for performance evaluation.

4.3.1. Evaluate in static network settings

Assuming a 128-node network, we evaluated the performance of the proposed algorithm in terms of assigning the variable number of internal data to the peers including 10, 20, 40, 80 data, which belong to four different datasets consisting of 1280, 2560, 5120 and 10,240 data, respectively. The algorithm behavior in static network setting is shown in Fig. 15(a)–(c).

Fig. 15a and b shows the evaluation of the GDSOM-P2P algorithm in the static network settings when number of internal data of peers varies from 10 to 40 and number of peers varies from 128 to 1024. According to the AC and RandI results, during 10 rounds of gossip interactions, the accuracy of data clustering results. After 6 rounds, rumor interactions reach 100% accuracy. The figure also shows that the algorithm is not sensitive to the number of internal data and network scalability when evaluated in terms of accuracy.

The average communication and storage overheads are shown when number of internal data of peers varies from 100 to 1000 for each node Fig. 15c.

Naturally, the amount of storage and communication overhead increases by increasing the number of internal data in each peer. However, the rising rate of storage overhead is manageable by the data summarization process as well as the communications office by age considerations for shared data.

4.3.2. Evaluate in dynamic network settings

Fig. 16a and b shows the evaluation of the GDSOM-P2P algorithm in a dynamic setting, with 10% churn, when number of internal data of peers varies from 10 to 40 and number of peers varies from 128 to 1024. According to the AC results, during 20 rounds of gossip interactions, the average accuracy of data clustering results for a network includes 128, 256, 512, and 1024 nodes is 96.04, 95.74, 97.31, and 96.88 respectively. In addition, the results of RandI are 98.33, 96.94, 98.01 and 97.81, respectively. This shows the efficiency and scalability of the algorithm in the dynamic network settings.

The average communication and storage overheads are shown when number of peers varies from 128 to 1024 in Fig. 16c. The communication and storage overheads of the algorithm remain constant due to the use of the age variable for shared data as well as the applying summarizing operations in each peer.

4.3.3. Comparison against LSP2P K-means and GD Clust

This section provides a comparison of the accuracy results for the proposed GDSOM-P2P algorithm with two of the state-of-the-art distributed data clustering techniques including the LSP2P K-means and the GDClust algorithm with two data-assignment strategies which aid at revealing special behaviors of the algorithms:

a: Random data assignment (RA): Each node is assigned data randomly chosen from D.

b: Cluster-aware data assignment (CA): Each node is assigned data from a limited number of clusters.

It should be noted that, in the LSP2P K-means method, the K-means initial centroids are identical in all nodes, which prohibit changing the number of produced clusters. In order to provide a fair comparison between these three algorithms, a small modification is applied to GDClust algorithm, where we assume that the initial K-means clustering centroids are available in all nodes, in GDClust. In addition, the number of final clusters is assumed constant, in all three algorithms. The proposed GDSOM-P2P algorithm overcomes above limitations and there is no need for predefining the fixed initial centroids as well as the number of centroids in all nodes. However, for a fair comparison with the LSP2P K-means and the GDClust algorithm, we assume the number of final clusters is constant in proposed GDSOM-P2P algorithm.

Fig. 17a and b shows the comparisons of our algorithm with GDClust and LSP2P in a static setting, against AC and RandI evaluation metrics. The results demonstrate that proposed GDSOM-P2P algorithm achieves higher AC and RandI values.

Fig. 18 shows the comparisons of proposed GDSOM-P2P algorithm with GDClust and LSP2P in a dynamic setting, in dynamic settings with RA strategy and churn ratio = 10%.
Fig. 15. Evaluation of GDSOM-P2P algorithm in dynamic network settings. (a) AC, (b) RandI, and (c) Communication and storage overhead.
Fig. 16. Evaluation of GDSOM-P2P algorithm in dynamic network settings. (a) AC, (b) RandI, and (c) Communication and Storage overhead.

The AC and RandI results in Fig. 18 demonstrate that the proposed GDSOM-P2P outperforms the LSP2P $K$-means and GD Cluster method. The communication overhead of all three algorithms increases when churn exists in dataset. The growth of communication overhead in LSP2P $K$-means is due to increased message passing for handling churn. In our algorithm GD Cluster method, this is due to increase in the amount of data exchanged between network nodes. However, in dynamic setting, our proposed method has a lower communication overhead than LSP2P $K$-means and competes with GD Cluster method.

5. Conclusions

According to unpredictable growth and dynamic nature of P2P networks, where data of peers are constantly changing, need to a continuously running algorithm is inevitable that can update
the clustering results efficiently. This paper proposes a novel distributed data clustering method by combining, an improved version of Silhouette algorithm, the Dynamic Self Organizing Maps neural network, and VICINITY protocol as a generic overlay management framework based on self-organization. The performance of the proposed GDSOM-P2P is evaluated using several different datasets and compared with two well-established distributed data clustering methods. The comparison demonstrates the improved quality of the clustering for the proposed GDSOM-P2P. To evaluate the clustering quality of our proposed method in a real network condition, PeerSim simulator is used as one of the most commonly used P2P simulator. The simulation results in the dynamic settings reveal that the proposed method can handle churn, without sensitively to the internal data and network scalability. The results also show good scalability of the proposed GDSOM-P2P with low communication overhead, without the need for the synchronization, search, and other complex processes.

In addition, an estimation method to determine the number of clusters and a novel centralized data clustering method based on the Dynamic Self Organizing Maps neural network and a topological structure is proposed for cases where the number is not specified in advance. The evaluation results show a significant accuracy improvement of the proposed TMB-Cluster algorithm. For future work, we plan to extend this approach for Social network-based P2P systems.

References


Rasool Azimi received his B.Sc. degree in Software Engineering from Mehran University, Guilan, Iran, in 2011 and the M.Sc. degree from Science and Research Branch, Islamic Azad University, Qazvin, Iran in 2014. His research interests include distributed data mining, data clustering, artificial intelligence, and their applications in power systems.

Hedieh Sajedi received a B.Sc. degree in Computer Engineering from AmirKabir University of Technology in 2003, and M.Sc and Ph.D degrees in Computer Engineering (Artificial Intelligence) from Sharif University of Technology, Tehran, Iran in 2006 and 2010, respectively. She is currently an Assistant Professor at the Department of Computer Science, University of Tehran, Iran. Her research interests include Image processing, Machine Learning, and Data Mining.