<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Incremental fuzzy clustering with multiple medoids for large data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Wang, Yangtao; Chen, Lihui; Mei, Jian-Ping</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Wang, Y., Chen, L., &amp; Mei, J.-P. (2014). Incremental fuzzy clustering with multiple medoids for large data. IEEE transactions on fuzzy systems, 22(6), 1557-1568.</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2014</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/25085">http://hdl.handle.net/10220/25085</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<a href="http://dx.doi.org/10.1109/TFUZZ.2014.2298244">http://dx.doi.org/10.1109/TFUZZ.2014.2298244</a>].</td>
</tr>
</tbody>
</table>
Incremental Fuzzy Clustering with Multiple Medoids for Large Data

Yangtao Wang, Lihui Chen, Senior Member, IEEE, and Jian-Ping Mei, Member, IEEE

Abstract—As an important technique of data analysis, clustering plays an important role in finding the underlying pattern structure embedded in the unlabelled data. Clustering algorithms that need to store the entire data into the memory for analysis become infeasible when the data set is too large to be stored. To handle such kind of large data, incremental clustering approaches are proposed. The key idea of these approaches is to find representatives (centroids or medoids) to represent each cluster in each data chunk, which is a packet of the data, and final data analysis is carried out based on those identified representatives from all the chunks. In this paper we propose a new incremental clustering approach called incremental multiple medoids based fuzzy clustering (IMMFC) to handle complex patterns that are not compact and well separated. We would like to investigate if IMMFC is a good alternative to capture the underlying data structure more accurately. IMMFC not only facilitates the selection of multiple medoids for each cluster in a data chunk, but also has the mechanism to make use of relationships among those identified medoids as side information to help the final data clustering process. The detailed problem formulation, updating rules derivation, and the in-depth analysis of the proposed IMMFC are provided. Experimental studies on several large data sets including real world malware data sets have been conducted. IMMFC outperforms existing incremental fuzzy clustering approaches in terms of clustering accuracy and robustness to the order of data. These results demonstrate the great potential of IMMFC for large data analysis.

Index Terms—Large data, incremental clustering, fuzzy clustering, multiple medoids, malware clustering.

I. INTRODUCTION

LARGE data becomes prevalent nowadays because huge amount of data can be collected easily every day from various sensors, social network sites, purchase transaction records, videos from surveillance systems to name a few. For example, 30 billion pieces of content are shared on Facebook every month, and 235 terabytes data is collected by the US Library of Congress by April 2011 [1]. According to Huber’s description [2], large data is defined as data with the size of $10^8$ bytes. Mining valuable information in the prevalent large data nowadays is crucial to get an edge of the competition for many parties including enterprises and government organizations. Clustering is a promising data analysis tool for finding the pattern structure and information underlining the unlabelled data. It has been widely studied in, to name a few, the areas of data mining, machine learning, information retrieval and cybersecurity. Different clustering algorithms based on various theories and applications have been developed over the past decades [3], [4], [5]. For large data clustering, one challenge is that the data is too large to be stored in the memory. Incremental clustering framework is adopted to handle this challenge by processing data chunk by chunk. One chunk is a packet or a small portion of the entire data. The chunk size can be decided by the users. In this framework, one representative object called medoid is identified for each cluster in a chunk. However, only one medoid may not be enough to capture the underlying structure of the data. In this paper, multiple medoids are considered for representing each cluster in a chunk to improve the performance of large data clustering.

Many clustering approaches have been designed to help address large data in the literature. For example in [6], CLARA is proposed to make use of the random sampling method to handle large data. In particular, it identifies the medoids of the samples to approximate the medoids of the entire data set. Each medoid is a prototypical object for representing the cluster. In [7], Ng et al. present CLARANS and the medoids are identified by building a graph structure and searching on it. Each node in the graph contains a set of medoids that represent the data set. Randomized search is used to determine the final set of the medoids. BIRCH [8] clusters data incrementally based on the in-memory balanced tree structure and summarization techniques. Since CLARA, CLARANS and BIRCH are unsuitable for non-spherical and unbalanced data, in [9], Guha et al. propose a hierarchical clustering method named CURE, which finds multiple representative objects in each cluster and employs random sampling method to handle large data. In [10], an incremental clustering algorithm for stream data based on k-Median called LSEARCH is developed using chunk-based processing style. They process data chunk by chunk and then each cluster in one chunk is represented by one weighted centroid. The LSEARCH is applied on all the centroids to get the final set of centroids for the entire data set. In [11], spectral clustering algorithm is extended to update the clustering results incrementally to handle the large evolving data. Recently, two approaches [12], [13] based on random sampling and random Fourier maps respectively are proposed for approximating kernel k means to solve large scale kernel clustering.

The clustering methods discussed above are all based on hard clustering which indicates whether an object belongs to a certain cluster or not. Many methods have been extended to soft or fuzzy clustering approaches in order to handle more real world data sets where data objects in those data sets may not be well separated. It has been discussed that soft clustering such as those popular ones in the literature [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] may capture the natural structure of a data set more closely. Each object in a data set may belong to all clusters with various
degrees of memberships. To handle large data, an online Nonnegative Matrix Factorization approach[24] is proposed for efficiently clustering large document data set. Several incremental fuzzy clustering algorithms based on the well known Fuzzy c means(FCM) [25] have been developed. The popular algorithms include the single-pass FCM referred as SPFCM [26] and online FCM called OFCM [27], [28]. The two algorithms process data chunk by chunk and estimate the c centroids for the entire data set by extracting the information in each chunk. The kernel versions of them called spkFCM and okFCM [29] are also developed for clustering large data efficiently. The SPFCM and OFCM are all based on a vector representation of data or be referred as object data. Recently incremental fuzzy clustering for relational data have also been proposed. In [30], two algorithms based on fuzzy c medoids(FCMD)[31] called online fuzzy c medoids(OFCMD) and history based online fuzzy c medoids(HOFCMD) are developed for clustering large relational data sets. However, as discussed in [32], one medoid may not be sufficient enough to capture the underlying structure of a cluster. In order to represent the structure of a cluster more accurately, multiple medoids may be a better choice.

Inspired by the above idea, we propose a new incremental fuzzy clustering approach called incremental multiple medoids based fuzzy clustering(IMMFC) for large relational data clustering. In IMMFC, the data is processed one chunk at a time instead of loading the entire data into the memory for analysis. Multiple medoids are identified for each cluster in a data chunk by introducing a weight for each object, which measures how well the object represents the cluster in a chunk. These medoids are used as the representatives of the data set to carry out the final data partition. We also propose the mechanism to make use of some relationships of identified medoids in each chunk as side information to guide the generation of the final set of medoids. In this paper, the detailed formulation, derivation and an in-depth analysis of the model are given. The experiments of IMMFC on several large data sets including two real world malware data sets show that IMMFC achieves better clustering accuracy and robustness to the order of the data than the related chunk based algorithms. Throughout this paper, the following denotations are used unless otherwise stated: an N-dimensional data set, $X$, whose relational matrix is denoted by $R_{n \times n}$, and letters in bold lowercase denote vectors, e.g., $w$, while bold $1_n$ is a vector with length $n$, in which all elements are 1s.

The rest of the paper is organized as follows: in the next section, a review on the related incremental fuzzy approaches reported in the literature is highlighted. In section 3, the details of the proposed incremental clustering approach IMMFC are presented. Experiments on several large data sets are conducted and the results are analyzed in section 4. Finally, conclusions are drawn in section 5.

II. RELATED WORK

In this section, four incremental fuzzy clustering algorithms recently proposed are reviewed. The common and unique characteristics of each of the four approaches are discussed.

A. SPFCM and OFCM

SPFCM[26] and OFCM[27] are two incremental fuzzy approaches designed based on FCM. To handle large data, SPFCM and OFCM use the same strategy which is to process the data chunk by chunk. For each chunk, a set of centroids is calculated to represent the chunk with one centroid per cluster. Instead of applying FCM directly on each chunk, weighted FCM(wFCM) is used for both approaches to determine the final set of centroids.

The significant difference of SPFCM and OFCM is the way of handling the centroids of each chunk. In SPFCM, the centroids identified from the previous chunk are combined into next chunk and the final set of centroids for the entire data is generated after last chunk is processed. While in OFCM, the identification of centroids for every chunk is processed individually and an additional step is needed to generate the final set of centroids for the entire data.

In SPFCM, the weight for the centroid of each cluster in each chunk is calculated as follows:

$$w_c = \sum_{i=1}^{n_p} (u_{ci})w_i, \quad 1 \leq c \leq k$$

Where, $w_c$ is the weight of centroid of the $c_{th}$ cluster, $n_p$ is the number of objects in $p_{th}$ chunk, $k$ is the number of clusters, $u_{ci}$ is the membership of object $i$ belongs to cluster $c$ and $w_i$ is the weight of object $i$. For the first chunk of the data ($p = 1$), $w_i$ is assigned to 1 for every object and $m = 0$. From the second chunk of data ($p \neq 1$), $m = k$ and the $k$ weighted centroids are combined with the $p_{th}$ chunk of data. The $n_p + k$ objects will be clustered by wFCM in which the weights of the $n_p$ objects in $p_{th}$ chunk are all set to 1 and the weights of $k$ objects are calculated from previous chunk. These steps continue until last chunk of data is processed.

In OFCM, the weight for the centroid of each cluster in each chunk is calculated as follows:

$$w_c = \sum_{i=1}^{n_p} (u_{ci})w_i, \quad 1 \leq c \leq k, \quad w_i = 1, \quad \forall 1 \leq i \leq n_p$$

The centroids are identified independently based on each chunk. The final set of centroids is generated by applying wFCM on the weighted centroids identified from all chunks.

As discussed above, both of the approaches handle large data by processing data chunk by chunk. The data type which the two approaches support is object data. However, if the input data is relational data, these approaches are not feasible. Next, we review two incremental fuzzy c medoids clustering algorithms which use similar mechanism as SPFCM and OFCM, but they are extended from handling the large object data sets to large relational data sets.

B. HOFCMD and OFCMD

In [30], two methods called HOFCMD and OFCMD are proposed for large relational data sets. Compared to SPFCM and OFCM, the major difference is the way they choose the representative of a cluster. In SPFCM and OFCM, the
representative which is called centroid is achieved by calculating the average value of the vector represented objects in a cluster. While in HOFCMD and OFCMD, the representative which is called medoid is achieved by selecting one of the real objects in the cluster. This difference makes SPFCM and OFCM always be applied to object data, and HOFCMD and OFCMD for relational data. In some applications, medoids based methods have an advantage of better interpretability by using medoids instead of centroids because medoids are real objects in the data set.

For handling large data, the incremental ways of HOFCMD and OFCMD are similar to SPFCM and OFCM respectively. Similar to wFCM, weighted fuzzy c medoids (wFCMD) is used and OFCMD are similar to SPFCM and OFCM respectively.

using medoids instead of centroids because medoids are real objects in the cluster. This difference makes SPFCM and OFCM always be applied to object data, and HOFCMD and OFCMD for relational data. In some applications, medoids based methods have an advantage of better interpretability by using medoids instead of centroids because medoids are real objects in the data set.

In HOFCMD, similar to SPFCM the identified weighted medoids in previous chunk are combined with current chunk as side information. Final set of medoids is generated after processing the last chunk of data. In OFCMD, similar to OFCM, the final set of medoids is generated by applying wFCMD on the medoids identified from all the chunks. The approach clusters all the chunks independently and aggregates all the medoids of the chunks in the end.

As discussed above, all the four methods use only one centroid or one medoid to represent each cluster. In our method, we identify multiple medoids to represent one cluster of each chunk and make use of some relationships of the medoids as side information to help the generation of the final medoids for the entire data set. Next, we propose our new incremental fuzzy clustering approach called IMMFC, including the detailed formulation, derivation and an in-depth analysis.

III. THE PROPOSED APPROACH

In this section, we first formulate the objective function of the proposed approach IMMFC integrating two types of mechanisms: selection of multiple medoids for each cluster and automatic collection of side information from identified medoids. The selection of multiple medoids is based on representative weight which reflects how well an object represents the cluster. The side information is collected in the form of a pair-wise constraints set. Each pair in the constraint set consists of two objects which have a high probability to be clustered in the same cluster. It is showed that the IMMFC clustering is actually an important optimization problem of the cost function with constraints. The updating rules are derived by applying the Lagrangian Multiplier method and Karush-Kuhn-Tucker (KKT) conditions. Next, we introduce the algorithm of IMMFC including the detail steps. The time complexity of the algorithm will be discussed as well.

In summary, multiple medoids are identified by IMMFC to represent each cluster in a chunk. Some relationships among those identified medoids are used to form pair-wise constraints as side information. The final set of medoids can be identified with the help of those chunk-medoids and their pair-wise constraints.

A. Objective function of IMMFC

First the following objective function is formulated for IMMFC $J_{IMMFC}$:

$$J_{IMMFC} = \sum_{c=1}^{k} \sum_{i=1}^{n} u_{ci} w_i \text{Dis}(x_i, \delta_c)$$

$$+ \frac{T_u}{2} \sum_{c=1}^{k} \sum_{i=1}^{n} (u_{ci})^2$$

$$+ \frac{T_v}{2} \sum_{c=1}^{k} \sum_{i=1}^{n} (v_{cj})^2 - S_u \sum_{x_m, x_n \in Q} \sum_{c=1}^{k} u_{cm} u_{cn}$$

subject to

$$u_{ci} = 1, \text{ for } i = 1, 2, ..., n$$

$$u_{ci} \geq 0, \text{ for } c = 1, 2, ..., k, i = 1, 2, ..., n$$

$$v_{cj} = 1, \text{ for } c = 1, 2, ..., k$$

$$v_{cj} \geq 0, \text{ for } c = 1, 2, ..., k, j = 1, 2, ..., n$$

In $J_{IMMFC}$, $r_{ij}$ is the distance or dissimilarity of object $i$ and object $j$, $u_{ci}$ is the fuzzy membership which represents the degree of object $i$ belongs to cluster $c$, $v_{cj}$ is a weight that reflects the degree of the representative of object $j$ for cluster $c$. The higher the value is, the better this object can represent this cluster, and we call it representative weight in this paper.

$J_{IMMFC}$ consists of four terms. The first term is the total dissimilarity within each cluster weighted by $u_{ci}$ and $v_{cj}$. The next two terms are the regularization terms which are used as penalties on the two fuzzy memberships. $T_u > 0$ and $T_v > 0$ are two parameters to control the weight of the two regularization terms in the objective function. The fourth term is the pairwise constraint term served as the side information to supervise the final clustering. In this term, $S_u > 0$ is the parameter to tune the weight of this term in the objective function. $Q$ denotes the set of pairs of objects in which each pair of objects has a high probability of being in the same cluster. The higher probability the objects in the pair are in the same cluster, the larger the value of term $\sum_{c=1}^{k} u_{cm} u_{cn}$.

When object $m$ and object $n$ are in the same cluster, the value of $\sum_{c=1}^{k} u_{cm} u_{cn}$ tends to be 1, which is the maximum value. In other words, the fourth term $\sum_{c=1}^{k} u_{cm} u_{cn}$ needs to be maximized in the objective function to ensure each pair of objects in set $Q$ is clustered into the same cluster.

The clustering task is to minimize the $J_{IMMFC}$ given set $Q$, and subject to the constraints in (5), (6), (7) and (8). Each
pair of constraint in set \( Q \) is constructed based on the representative weight of medoids identified from each chunk. In this work, the pairs of constraints are generated automatically after clustering each chunk. We treat the clustering process as solving the optimization problem of the cost function with constraints. By applying the Lagrangian Multiplier method and KKT conditions, we derive the membership updating rules for the new IMMFC. The Lagrangian function considering the constraints is given as follows:

\[
L_{IMMFC} = J_{IMMFC} + \sum_{i=1}^{n} \lambda_i(\sum_{c=1}^{k} u_{ci} - 1) + \sum_{j=1}^{k} \beta_j(\sum_{i=1}^{n} v_{cj} - 1) + \sum_{c=1}^{k} \sum_{i=1}^{n} \psi_{ci} u_{ci} + \sum_{c=1}^{k} \sum_{j=1}^{n} \phi_{cj} v_{cj}
\]

where the \( \lambda_i \), \( \beta_j \), \( \psi_{ci} \) and \( \phi_{cj} \) are the Lagrange multipliers, and the KKT conditions for solving \( u_{ci} \) are given as follows:

\[
\frac{\partial L_{IMMFC}}{\partial u_{ci}} = 0, \quad \psi_{ci} \geq 0, \quad \psi_{ci} u_{ci} = 0.
\]

The updating rule of \( u_{ci} \) can be derived as

\[
u_{ci} = \begin{cases} 0 & \text{for } c \in q^- \\ \frac{S_u(u_{ci} + \frac{1}{|q^-|} \sum_{f \in q^-} u_{fi} - \frac{1}{|q^-|})}{T_u} & \text{for } c \in q^+ \\ \frac{1}{|q^+|} - \frac{1}{T_u} \left( \sum_{j=1}^{n} v_{cj} r_{ij} - \frac{1}{|q^+|} \sum_{f \in q^+} \sum_{j=1}^{n} v_{fj} r_{ij} \right) & \text{for } c \in q^+ \end{cases}
\]

where \( u_{ch} \) is the membership of object \( h \) to cluster \( c \). Object \( h \) and object \( i \) is a pair of constraint in set \( Q \), and

\[
q^- = \{ c : u_{ci} = 0 \} \\
q^+ = \{ c : u_{ci} > 0 \}
\]

By using the similar derivation method, the updating rule of \( v_{cj} \) is given as follows:

\[
v_{cj} = \begin{cases} 0 & \text{for } j \in p^- \\ \frac{1}{|p^+|} - \frac{1}{T_v} \left( \sum_{i=1}^{n} u_{ci} r_{ij} - \frac{1}{|p^+|} \sum_{j=1}^{n} u_{fj} r_{ij} \right) & \text{for } j \in p^+ \end{cases}
\]

where

\[
p^- = \{ j : v_{cj} = 0 \} \\
p^+ = \{ j : v_{cj} > 0 \}
\]

Here, \( p^- \) and \( p^+ \) are the sets of objects with zero representative weight for cluster \( c \) and greater than zero representative weight for cluster \( c \) respectively. \( |p^-| \) and \( |p^+| \) represent the number of objects in set \( p^- \) and \( p^+ \) respectively. These two sets are determined by using the similar method as that of \( q^- \) and \( q^+ \).

The procedure for determining them is outlined as follows.

**Procedure for determining \( p^- \) and \( p^+ \)**

1. Initialize \( p_0^+ = \emptyset, p_0^- = 1, 2, \ldots, n, s = 0; \)
2. \( s \leftarrow s + 1, p_s^+ = p_{s-1}^+ + \{m\}, p_s^- = p_{s-1}^- - \{m\}, \)
   where \( m = \arg \min_{c \in q^-} \{ \sum_{i=1}^{n} u_{ci} r_{ij} \}; \)
3. Check whether \( v_{cj} > 0 \) computed by Eq. 15,
   where \( g = \arg \max_{j \in p^+} \{ \sum_{i=1}^{n} u_{ci} r_{ij} \}. \)
   If yes, go to step 2,
   else set \( p^+ = p_{s-1}^+; p^- = p_{s-1}^- \) and terminate.

Next, we propose the details of incremental fuzzy clustering algorithm IMMFC.

**B. IMMFC Algorithm**

The proposed IMMFC processes each chunk of data individually. The final set of medoids can be identified from the chunk-medoids with their pair constraints as the side information.

The main differences between IMMFC and the methods reported in the literature are (1) a new mechanism is provided to select multiple medoids instead of a single one to represent each of the clusters in each chunk, (2) both the fuzzy membership and representative weight of objects in the objective function are updated iteratively in IMMFC, (3) the auto-generated pairwise constraints from those identified medoids are used as the side information for determining the final set of medoids. The IMMFC approach is outlined as follows.

First, data set \( X \) is partitioned into non-overlapping partitions of \( X = \{ X_1, X_2, \ldots, X_P \} \). The number of objects in each chunk \( X_p \) is \( n_p \). As showed in the following algorithm, \( R_{n_p \times n_p}^p \) is the distance matrix of chunk \( p \) which is calculated based on the data in the \( p \)th chunk. In step 1, medoids set \( M \) and pairwise constraints set \( Q \) are initialized. In step 2, the fuzzy membership matrix \( U_p \) and representative weight matrix \( V_p \) for each chunk \( p \) are calculated by using Algorithm-1 with \( S_u = 0 \). In Algorithm-1, \( U_p \) is initialized firstly. \( V_p \) and \( U_p \) are alternatively updated by using (15) and (13) respectively.
until the difference between successive estimation of $U_p$ is small enough. $v^0_{cj}$, which is the element of matrix $V_p$ is the representative weight of object $j$ for cluster $c$ of chunk $p$. Then, $t$ objects with most representative weight are identified to be medoids for cluster $c$ of chunk $p$. The two medoids with top 2 representative weight are taken to form a pair of constraint automatically and added into set $Q$. In step 3, distance matrix $R_{|M|×|M|}$ is calculated from medoids set $M$. We then calculate the membership matrix $U$ and $V$ by using Algorithm-1 with a suitable $S_u \neq 0$ and set $Q \neq \emptyset$. The final medoids set $M^f$ is identified in step 4 in which the object in each cluster with the highest representative weight is selected as the medoid. In the final step, cluster indicator $q$ is determined for each object. $q_j$ is the cluster number which object $j$ belongs to. This is acquired by assigning object $j$ to the cluster whose medoid is nearest to object $j$.

It can be seen that some parameters need to be set before running the algorithm. It includes the stopping criterion $\epsilon$, chunk size $n_p$, parameters $T_u$, $T_v$ and $S_u$, number of clusters $k$, and number of medoids for every cluster $t$. The stopping criterion $\epsilon$ and chunk size $n_p$ are set by user according to the data set. $T_u$ and $T_v$ are the parameters that control the contribution of regularization terms into the objective function. $S_u$ controls the weight of pairwise constraints term in the objective function. For tuning the three parameters, we give the guideline to set $T_u = T_u n_p / k$ and $S_u = T_u N / (2q)$ to make the three terms have the same magnitude. Here, $N$ is the number of medoids identified from all the chunks and $q$ is the number of pair of constraints in set $Q$. Therefore, $T_u$ can be tuned manually according to the specific data set and $T_u, S_u$ can be calculated accordingly. About setting the number of clusters for every chunk, we applied the same principle as discussed in [27]. Suppose the number of classes of each chunk may be less than $k$. However, the overclustering unlikely causes information loss [27]. Therefore we set the number of clusters to be k-the number of classes in the data set for each chunk.

The time complexity of IMMFC is $O(P(n_p^2 + t \cdot k) + (|M|^2 + k))$ considering the chunk number $P$ and cluster number $k$. $O(P(n_p^2 + t \cdot k))$ is the time complexity of processing $P$ chunks and identifying $t$ medoids for each cluster. $O(|M|^2 + k)$ is the time complexity of the final clustering where $|M|$ is the number of identified medoids from all the chunks. In real application, $t \cdot k$ is always much less than $n_p^2$ and $k$ is much less than $|M|^2$. Therefore the time complexity can be considered as $O(P \cdot n_p^2 + |M|^2)$. To speed up the computation, we can set the elements in distance matrix with small value to 0 or using sampling methods as reported in [33]. Next, we apply IMMFC on a synthetic example to illustrate the working principle of the algorithm. It provides us some insights into IMMFC.

**Algorithm: IMMFC**

**Input:** Distance matrix of $p$th chunk $R_{n_p \times n_p}$, number of clusters $k$, number of medoids for every cluster $t$, parameter $T_u$, $T_v$ and $S_u$, stopping criterion $\epsilon$

**Output:** Cluster Indicator $q$

**Method:**
1. Initialize Medoids set $M = \emptyset$, Pairwise constraints set $Q = \emptyset$.
2. for $p = 1$ to $P$
   - Calculate $U_p, V_p$ using Algorithm-1 with input parameter $S_u = 0$
   - for $c = 1$ to $k$
     - $s = 0$, set $A = 1, 2, \ldots n_p$;
     - Repeat
       - $s = s + 1$
       - $z = \arg \max_{j \in A} v^0_{cj}$; medoid: $m_x = x_z$
       - $M = M \cup \{m_x\}$;
       - $A = A \setminus \{z\}$;
   - Until(s=\text{t})
   - Take the two most representative medoids for cluster $c$ to form a pairwise constraint $(m_1, m_2)$ and add it into set $Q$: $Q = Q \cup \{(m_1, m_2)\}$
3. Calculate $U, V$ using Algorithm-1 with input parameter $S_u > 0$ based on $R_{|M| \times |M|}$ and pairwise constraints set $Q$
4. Final medoids set $M^f = \emptyset, c = 0$
   - Repeat
     - $c = c + 1$
     - $z = \arg \max_{1 \leq j \leq |M|} v_{cj}$; medoid: $m_c = x_z$
     - $M^f = M^f \cup \{m_c\}$;
   - Until(c=k)
5. $q_j = \min_{1 \leq c \leq k} \text{Dis}(m_c, x_j)$,
   - $j = 1, 2, \ldots n$, $n = \sum_{p=1}^{P} n_p$

**Algorithm-1:**

**Input:** Distance matrix $R_{n \times n}$, number of clusters $k$, parameter $T_u$, $T_v$, $S_u$, stopping criterion $\epsilon$

**Output:** Fuzzy membership matrix $U$, representative weight matrix $V$

**Method:**
1. Initialize $U_p^0$, set iteration number $l \leftarrow 0$
2. Repeat
   - $l \leftarrow l + 1$
   - Update $V_p^l$ using (15) based on $U_p^{l-1}$
   - Update $U_p^l$ using (13) based on $V_p^l$
   - Until($\|U_p^l - U_p^{l-1}\| < \epsilon)$

**C. Synthetic example**

We apply IMMFC on the 2D15 (n=5000, k=15, d=2) data set to show the process of the algorithm. 2D15 is a synthetic data set which is composed of 5000 two dimensional points with 15 classes. The distribution of the points is showed in

---

1. This data set was designed by Ilia Sidoroff and can be downloaded on http://www.uef.fi/en/sipu/datasets.
Fig. 1: 2D15 synthetic data

Fig. 1. The data is considered as coming in four chunks randomly. For each cluster we choose two medoids. The process of algorithm is showed in Fig. 2. The data distribution of the first chunk is showed in Fig. 2(a), the black pentagrams represent the medoids for each cluster. It can be seen in the figure that the two selected medoids are in the central area of each cluster. Then the same procedures are conducted for the other three chunks as showed in Fig. 2(b), (d) and (e) respectively. Pairwise constraints set consists of all the medoids. Each two selected medoids forms one pair of constraint in the set. Then all the medoids identified from each chunk are used as the objects with the help of side information given in the set to find the final medoids for the entire data set which is showed in Fig. 2(c). It is noted that the final medoids founded are located in the ideal position in the data set. The evaluation of the clustering results will be showed in next section.

IV. EXPERIMENTAL RESULTS

In this section, experimental studies of the proposed approach are conducted on five data sets including two real world malware data sets. In the experiments, we compare the performance of IMMFC with OFCMD and HOFCMD to show the clustering results and robustness to the order of the data. For interesting application in software security, two malware data sets are used for the study. Three types of experiments are conducted and reported. First, we compare IMMFC with OFCMD and HOFCMD to see if using multiple medoids to represent each chunk improves the performance of clustering. Second, other than the two incremental fuzzy approach, we also compare IMMFC with our previous work PFC[32] which is a non-incremental fuzzy approach considering multiple medoids to see if the proposed IMMFC is better than PFC. Third, the experiments of parameter analysis are conducted with various values to show the impact of parameters. The experiments implemented in Matlab were conducted on a PC with four cores of Intel I5-2400 with 24 gigabytes of memory.

A. Data sets

We compare the performance of the algorithms on the following data sets.

2D15: The characteristic of this synthetic data is given in the above section.

MNIST\(^2\): This data set is composed of 10 classes which are 0 to 9 handwritten digit images. There are 70000 28 × 28 pixel images. We normalize the pixel value to [0,1] by dividing 255 and each image is represented as a 784 dimensional feature vector.

Forest\(^3\): This data set is from United States Geological Survey and United State Forest Service and it has 7 classes, 581012 objects. Each object is represented as a 54 dimensional feature vector.

Malware24: For malware data, it is referred as Malware 24 since there are 24 classes and the total number of objects is 3131. The data set is acquired from [34] whose general characteristic is showed in Table. I. The class size of the data set ranges from 26 to 300. The feature is extracted by Malheur based on the MIST representation[34] of malware behavior.

Malware28: A larger malware data set which we refer as Malware 28 is created based on the above malware data by adding 4 new malware classes. The detail of this data set is showed in Table. II. There are 28 malware classes with class size ranging from 26 to 1575 and the total number of objects is 6140.

<table>
<thead>
<tr>
<th>TABLE I: Malware24 data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware class</td>
</tr>
<tr>
<td>Adultbrowser</td>
</tr>
<tr>
<td>Allaple</td>
</tr>
<tr>
<td>Bancos</td>
</tr>
<tr>
<td>Casino</td>
</tr>
<tr>
<td>Dorfdn</td>
</tr>
<tr>
<td>Ejik</td>
</tr>
<tr>
<td>Flystudio</td>
</tr>
<tr>
<td>Lipinch</td>
</tr>
<tr>
<td>Looper</td>
</tr>
<tr>
<td>Magiccasino</td>
</tr>
<tr>
<td>Podnuka</td>
</tr>
<tr>
<td>Posion</td>
</tr>
<tr>
<td>Posion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II: Malware28 data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware class</td>
</tr>
<tr>
<td>Adultbrowser</td>
</tr>
<tr>
<td>Allaple</td>
</tr>
<tr>
<td>AutoIt</td>
</tr>
<tr>
<td>AutoRun</td>
</tr>
<tr>
<td>Bancos</td>
</tr>
<tr>
<td>Basun</td>
</tr>
<tr>
<td>Casino</td>
</tr>
<tr>
<td>Dorfdn</td>
</tr>
<tr>
<td>Ejik</td>
</tr>
<tr>
<td>Flystudio</td>
</tr>
<tr>
<td>Lipinch</td>
</tr>
<tr>
<td>Looper</td>
</tr>
<tr>
<td>Magiccasino</td>
</tr>
<tr>
<td>Podnuka</td>
</tr>
</tbody>
</table>

\(^2\)This data set can be downloaded on http://yann.lecun.com/exdb/mnist/.

\(^3\)This data set can be downloaded on http://uisacad2.uis.edu/dstar/data/clusteringdata.html.
B. Evaluation criterion

Three popular external criterions Accuracy [17], F-measure [35], and Normalized Mutual Information (NMI) [36] are used to evaluate the clustering results, which measure the agreement of the clustering results produced by an algorithm and the ground truth. If we refer class as the ground truth, and cluster as the results of a clustering algorithm, the NMI is calculated as follows:

\[
NMI = \frac{\sum_{c=1}^{k} \sum_{p=1}^{m} n_c^p \log \left( \frac{n_c^p n_c}{n_c n_p} \right)}{\sqrt{\left( \sum_{c=1}^{k} n_c \log \left( \frac{n_c}{n} \right) \right) \left( \sum_{p=1}^{m} n_p \log \left( \frac{n_p}{n} \right) \right)}}
\]

(17)

where \(n\) is the total number of objects, \(n_c\) and \(n_p\) are the numbers of objects in the \(c_{th}\) cluster and the \(p_{th}\) class, respectively, and \(n_c^p\) is the number of common objects in class \(p\) and cluster \(c\). For F-measure, the calculation based on precision and recall is as follows:

\[
F\text{-measure} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

(18)

where,

\[
\text{precision} = \frac{n_c^p}{n_c}
\]

(19)

\[
\text{recall} = \frac{n_c^p}{n_p}
\]

(20)

Accuracy is calculated as follows after obtaining a one-to-one match between clusters and classes:

\[
\text{Accuracy} = \sum_{c=1}^{k} \frac{n_c^2}{n}
\]

(21)

where \(n_c^2\) is the number of common objects in the \(c_{th}\) cluster and its matched class \(j\). The higher the values of the three criterions are, the better the clustering result is. The value is equal to 1 only when the clustering result is same as the ground truth.

C. Initialization

For OFCMD and HOFCMD, we initialize the medoids by using the method adopted in [31]. The object which has the minimum distance to all the other objects is selected as the first medoid. The remaining medoids are chosen consecutively by selecting the objects that maximize their minimal distance with existing medoids. This helps the medoids distribute evenly in the data space to avoid converging to a bad local optimum. For the proposed IMMFC, we initialize the input fuzzy membership \(U\) by using similar method. First, we choose the medoids with the same method mentioned above. Then, if the medoid for cluster \(c\) is object \(i\), we set \(u_{ci} = 1\), and set the membership of the other objects to the same cluster as 0. The detail steps of initialization of IMMFC are as follows.

D. Results on 2D15, MNIST and Forest data sets

The parameter \(T_u\) is set to be 0.03, 0.1, 0.1 for 2D15, MNIST, and Forest data set respectively, and \(T_v, S_u\) are calculated according to the guideline. We set number of medoids for each cluster \(t = 2\), and \(\epsilon = 10^{-5}\) for IMMFC. We set fuzzifier \(m = 1.7\) which is recommended in [29] for OFCMD and HOFCMD to make them work well for these data sets. Number of clusters \(k\) is set to be 2, 10, 7 for 2D15, MNIST and Forest data sets respectively. The entire data set is partitioned randomly into equal sized chunks. The size of each chunk is
**Initialization for IMMFC**

Set the number of clusters \( k \),

fuzzy membership matrix is \( U_{k \times n} \)

Calculate the first medoid:

\[
p = \arg \min_{1 \leq j \leq n} \sum_{i=1}^{n} \text{Dis}(x_i, x_j);
\]

first medoid: \( v_1 = x_p \)

Medoids set \( V = \{ v_1 \}, m = 1 \);

Fuzzy membership \( u_{mp} = 1 \), and \( u_{mj} = 0 \), with \( j = 1, 2 \ldots n, j \neq p \)

**Repeat**

\[
m = m + 1
\]

\[
p = \arg \max_{1 \leq i \leq n, x_i \in V} \min_{1 \leq k \leq \| V \|} \text{Dis}(v_k, x_i);
\]

medoid: \( v_m = x_p \)

\[
V = V \cup \{ v_m \};
\]

\( u_{mp} = 1 \), and \( u_{mj} = 0 \), with \( j = 1, 2 \ldots n, j \neq p \)

**Until**(m=k)

decided by the user. Normally it refers to a certain percentage of the entire data set size. The size of the last chunk maybe smaller than the others if the entire data set can not be divided by the chunk size. For 2D15 and MNIST data sets, we conduct experiments with chunk sizes as 1%, 2.5%, 5%, 10% and 25% of the entire data set size. For Forest data set, limited by memory, smaller percentages are chosen for chunk sizes as 0.1%, 0.25%, 0.5%, 1% and 2.5% of the entire data set size.

To see the robustness of each approach, each of them runs 20 trials with random order of the input data. We calculate the mean, standard deviation, minimum and maximum of the values of accuracy, F-measure and NMI over 20 trials. The mean value reflects the average performance of the clustering algorithm. The standard deviation, minimum and maximum values reflect the robustness of the clustering algorithm to the order of the data. The results of the three data sets are showed in Table. III, Table. IV and Table. V respectively. From the tables we can see, IMMFC always produces the best partition every time with various chunk sizes. The results also show that IMMFC always has the lowest standard deviation over 20 trials which reflects that IMMFC is more robust to the order of the data sequence.

Table. III (a), (b), (c) are the accuracy results on 2D15, MNIST and Forest data set respectively. The results show that HofCMD performs better than OFCMD when the chunk size is small, and OFCMD performs better than HOFCMD when the chunk size increases to a certain percentage. IMMFC performs better not only on the accuracy, but also on the robustness to the order of data and different chunk sizes. The accuracy results on MNIST and Forest data sets provide similar pattern in performance although the accuracy value becomes lower than the value on 2D15. This phenomenon was also observed in related work [29] by using the same data set. Compared with OFCMD on the three data sets, the improvements of IMMFC on average accuracy over all the chunk sizes are 3.2%, 15% and 4.3%. Compared with HOFCMD, the improvements of IMMFC are 3.9%, 57.8% and 16.1%. The results of F-measure and NMI given in Table. IV and Table. V show similar pattern as accuracy.

### Table III: Accuracy of OFCMD, HOFCMD and IMMFC

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>OFCMD</th>
<th>HOFCMD</th>
<th>IMMFC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2D15</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chunk size</td>
<td>avg. std.</td>
<td>avg. std.</td>
<td>avg. std.</td>
</tr>
<tr>
<td>1%</td>
<td>0.9315 0.0432</td>
<td>0.9766 0.0338</td>
<td>0.9810 0.0252</td>
</tr>
<tr>
<td>2.5%</td>
<td>0.9308 0.0340</td>
<td>0.9841 0.0248</td>
<td>0.9933 0.0002</td>
</tr>
<tr>
<td>5%</td>
<td>0.9741 0.0319</td>
<td>0.9554 0.0391</td>
<td>0.9932 0.0003</td>
</tr>
<tr>
<td>10%</td>
<td>0.9792 0.0301</td>
<td>0.9253 0.0323</td>
<td>0.9932 0.0003</td>
</tr>
<tr>
<td>25%</td>
<td>0.9834 0.0220</td>
<td>0.9270 0.0301</td>
<td>0.9924 0.0012</td>
</tr>
<tr>
<td><strong>MNIST</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chunk size</td>
<td>avg. std.</td>
<td>avg. std.</td>
<td>avg. std.</td>
</tr>
<tr>
<td>1%</td>
<td>0.3334 0.0565</td>
<td>0.2974 0.0564</td>
<td>0.4448 0.0322</td>
</tr>
<tr>
<td>2.5%</td>
<td>0.2919 0.1412</td>
<td>0.2129 0.3916</td>
<td>0.4106 0.5009</td>
</tr>
<tr>
<td>5%</td>
<td>0.4069 0.0556</td>
<td>0.2687 0.0309</td>
<td>0.4352 0.0260</td>
</tr>
<tr>
<td>10%</td>
<td>0.3157 0.4758</td>
<td>0.2140 0.3075</td>
<td>0.4121 0.4740</td>
</tr>
<tr>
<td>25%</td>
<td>0.4123 0.0498</td>
<td>0.2930 0.0711</td>
<td>0.4454 0.0311</td>
</tr>
<tr>
<td><strong>Forest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chunk size</td>
<td>avg. std.</td>
<td>avg. std.</td>
<td>avg. std.</td>
</tr>
<tr>
<td>1%</td>
<td>0.3570 0.0241</td>
<td>0.3181 0.0340</td>
<td>0.3873 0.0154</td>
</tr>
<tr>
<td>0.25%</td>
<td>0.3836 0.0248</td>
<td>0.3348 0.0452</td>
<td>0.4058 0.0185</td>
</tr>
<tr>
<td>0.5%</td>
<td>0.3429 0.4100</td>
<td>0.2698 0.3847</td>
<td>0.3727 0.4157</td>
</tr>
<tr>
<td>1%</td>
<td>0.3760 0.0565</td>
<td>0.3192 0.0364</td>
<td>0.3941 0.0248</td>
</tr>
<tr>
<td>2.5%</td>
<td>0.3830 0.0171</td>
<td>0.3410 0.0364</td>
<td>0.3959 0.0224</td>
</tr>
</tbody>
</table>
| **E. Results on Malware24**

For IMMFC, we set the parameters \( T_u = 0.1 \), number of clusters \( k = 24 \), number of medoids for each cluster \( t = 2 \), and \( \epsilon = 10^{-5} \). \( T_v \) and \( S_u \) are set according to the guideline. And we set fuzzifier \( m = 1.7 \), and number of clusters \( k = 24 \) for OFCMD and HOFCMD. Each approach is run for 20 trials. The chunk sizes of all the approaches are selected as 1%, 2.5%, 5%, 10% and 25% of the entire data set. The best results of each approach with respect to different chunk sizes are tabulated in Table. VI. It can be seen that IMMFC yields higher accuracy, NMI and F-measure values than OFCMD and HOFCMD. The results show that the mechanism using multiple medoids to represent each chunk and using side information in the final labeling is more effective than the one using single medoid. Also from the standard deviation value in Table. VI, we can see that IMMFC is more robust.
TABLE IV: F-measure of OFCMD, HOFCMD and IMMFC

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>OFCMD</th>
<th>HOFCMD</th>
<th>IMMFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>0.5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>1%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>2.5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>10%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>25%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
</tbody>
</table>

TABLE V: NMI of OFCMD, HOFCMD and IMMFC

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>OFCMD</th>
<th>HOFCMD</th>
<th>IMMFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>0.5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>1%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>2.5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>5%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>10%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
<tr>
<td>25%</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
<td>0.901 0.042</td>
</tr>
</tbody>
</table>

TABLE VI: Comparison of OFCMD, HOFCMD and IMMFC on Malware24

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>OFCMD</th>
<th>HOFCMD</th>
<th>IMMFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.9600 ± 0.0084</td>
<td>0.9600 ± 0.0084</td>
<td>0.9600 ± 0.0084</td>
</tr>
<tr>
<td>NMI</td>
<td>0.9851 ± 0.0004</td>
<td>0.9851 ± 0.0004</td>
<td>0.9851 ± 0.0004</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.8454 ± 0.0123</td>
<td>0.8454 ± 0.0123</td>
<td>0.8454 ± 0.0123</td>
</tr>
</tbody>
</table>

F. Results on Malware28

For Malware28 data set, we set the parameters for each algorithm the same as the setting in the above section. Table VII shows best results of accuracy, NMI and F-measure of OFCMD, HOFCMD and IMMFC with respect to different chunk sizes. In each trial, the order of data is randomized. From the mean value and standard deviation of the three criterions, it can be seen that IMMFC still performs better than OFCMD and HOFCMD on the larger malware data set with more accurate clustering performance and more robust to the order of the data.

G. Compare with PFC

We also compared the IMMFC with our previous work PFC, which is the only existing non-incremental and multiple medoids fuzzy clustering approach for relational data in the literature. We wish to see if the incremental approach causes
any obvious trade off in terms of clustering accuracy when compared with the non-incremental approach using entire data set.

For both IMMFC and PFC, the simulations are run under the same parameter setting and using the same initialization method as in the above section. Since PFC is non-incremental approach which needs to store the entire distance matrix in the memory, we are only able to get PFC results on 4 out of 5 data sets with our current computer configuration (RAM: 24G), limited by the memory size. IMMFC is run for 20 trials and the mean values of the three evaluation criterions are used to compare with PFC. Because the result of every trial of PFC is the same when the initialization method is used, PFC only needs to be run once. The best results of both approaches are tabulated in Table VIII, where the chunk size of IMMFC needs to be run once. The best results of both approaches is the same when the initialization method is used, PFC only to compare with PFC. Because the result of every trial of PFC and the mean values of the three evaluation criterions are used of 5 data sets with our current computer configuration (RAM: 24G).

Table VII: Comparison of OFCMD, HOFCMD and IMMFC on Malware28

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>Accuracy</th>
<th>NMI</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFCMD</td>
<td>0.8007 ± 0.0603</td>
<td>0.8818 ± 0.0225</td>
<td>0.7934 ± 0.0418</td>
</tr>
<tr>
<td>HOFCMD</td>
<td>0.8040 ± 0.0380</td>
<td>0.8983 ± 0.0140</td>
<td>0.8055 ± 0.0300</td>
</tr>
<tr>
<td>IMMFC</td>
<td>0.9107 ± 0.0267</td>
<td>0.9431 ± 0.0143</td>
<td>0.9060 ± 0.0290</td>
</tr>
</tbody>
</table>

H. Parameter Analysis

To show both the effectiveness of the guideline and the impact of the parameters on the performance of IMMFC, we set $T_u$ in different values as: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3 and 1. $T_s$ and $S_u$ are calculated according to the guideline discussed in section III B. The following Fig. 3 shows the accuracy results of IMMFC on the 2D15, MNIST and Forest with chunk sizes 5%, 5%, 0.5% respectively. Here we only show the accuracy results, the results of F-measure and NMI have the similar pattern. From Fig. 3 we notice that the parameter $T_u$ and its corresponding $T_s$ and $S_u$ can significantly affect the performance of IMMFC. As given in Fig. 3 (a), (b), (c), when $T_u$ is set to be 0.03, 0.1 and 0.1 respectively, IMMFC achieves the best performance. Generally, very small or very large value of $T_u$ will lead to degenerated performance of IMMFC. Therefore, we recommend to tune $T_u$ between 0.01 and 0.3 in practice.

V. Conclusion

We have proposed a new incremental fuzzy clustering approach called IMMFC for large data analysis, and apply IMMFC on large image data and malware data to demonstrate its feasibility and potential. IMMFC processes large data chunk by chunk. One distinctive characteristic of IMMFC is that the mechanism is provided to capture and represent each of the clusters in a data chunk by multiple medoids not a single medoid. The other characteristic is that some pairwise constraints are automatically generated from the medoids identified throughout every chunk and used to help the final data partition. Experiments conducted on a synthetic data set, and several large data sets including two real world malware data sets show IMMFC outperforms related incremental algorithms with more accurate clustering results and more robust to the order of data. Moreover, IMMFC achieves higher or comparable clustering results compared with the related non-incremental algorithm. The merits showed in the experiments indicate that IMMFC has a great potential to be used for large data clustering. In the future, much larger data sets need to be collected and tested. Further experimental studies may be conducted to investigate the most appropriate number of medoids for a specific data set.

APPENDIX A

Derivation of Updating Rules

The detail derivation of fuzzy membership matrix U and representative matrix V are given here based on the Lagrangian function in (9) and KKT conditions in (10), (11), and (12). Derived by (10) and (5),

\[ u_{ci} = \frac{S_u (u_{ch} - \frac{1}{k}) + \frac{1}{k} \left[ (\psi_{ci} + \alpha_{ci}) - \frac{1}{k} \sum_{f=1}^{k} (\psi_{fi} + \alpha_{fi}) \right]}{T_u} \]

where

\[ \alpha_{ci} = \sum_{j=1}^{n} v_{cj} r_{ij}, \]

We consider the condition in (11) from two cases respectively.

1) $\psi_{ci} = 0$, $\forall c \in 1, 2...k$, then (22) becomes

\[ u_{ci} = \frac{S_u (u_{ch} - \frac{1}{k}) + \frac{1}{k} \left[ \psi_{ci} - \frac{1}{k} \sum_{f=1}^{k} \psi_{fi} \right]}{T_u} \]

For every object, this set of solution is only valid if

\[ \forall c \in 1, 2...k, S_u (u_{ch} - \frac{1}{k}) + \frac{1}{k} \left[ \psi_{ci} - \frac{1}{k} \sum_{f=1}^{k} \psi_{fi} \right] \geq 0, \]

otherwise, the next case need to be considered.

2) $\psi_{ci} \geq 0$ for at least one $c$. According to KKT condition (12), when $\psi_{ci} > 0$, $u_{ci} = 0$, and when $u_{ci} > 0$, $\psi_{ci} = 0$. Therefore, we split the set of clusters to two subsets named $q^-$ and $q^+$ according to constraint in (6), with

\[ q^- = \{ c : u_{ci} = 0 \} \]
\[ q^+ = \{ c : u_{ci} > 0 \} \]

so for $c \in q^-$, $u_{ci} = 0$, and we need derive $u_{ci}$ when $c \in q^+$. Because $c \in q^+$, $u_{ci} > 0$, then $\psi_{ci} = 0$, and
(22) is rewritten as:

\[
    u_{ci} = \frac{S_u}{T_u} (u_{ch} - \frac{1}{k}) + \frac{1}{k} - \frac{1}{T_u} \alpha_{ci} + \frac{1}{T_u} k \left[ \sum_{f \in q^+} \alpha_{fi} + \sum_{f \in q^-} (\alpha_{fi} + \psi_{fi}) \right],
\]

(27)

for \( f \in q^-, \ u_{ci} = 0 \), according to (22), we can get:

\[
    \alpha_{fi} + \psi_{fi} = S_u (u_{ch} - \frac{1}{k}) + \frac{T_u}{k} + \frac{1}{k} \sum_{m=1}^k (\alpha_{mi} + \psi_{mi})
\]

(28)

Then, after several steps of derivation, the following equation is obtained:

\[
    \sum_{f \in q^-} (\alpha_{fi} + \psi_{fi}) = \frac{kS_u}{|q^-|} \sum_{f \in q^-} u_{fh} - \frac{S_u}{|q^-|} |q^-| + \frac{|q^-|}{|q^-|} \sum_{f \in q^-} \alpha_{fi} + \psi_{fi}
\]

(29)

Substituting (29) to (27), we finally obtain the update equation for \( u_{ci} \) as showed in (13).

The updating equation of \( V \) as in (15) can also be derived using similar procedure of derivation.

REFERENCES
