Using M Tree Data Structure as Unsupervised Classification Method

Marian Cristian Mihăescu and Dumitru Dan Burdescu
University of Craiova, Faculty of Automation Computers and Electronics, Romania
E-mail: mihaescu@software.ucv.ro, burdescu@software.ucv.ro

Keywords: M Tree, k-means, unsupervised classification, cognitonics

Received: December 8, 2011

Increasing the effectiveness of educational processes is one of the greatest challenges for information society. The paper presents the usage of M Tree structure for classification of the learners based on their final marks obtained in their respective courses. The classical building algorithm of M-Trees with an original accustomed clustering procedure was implemented. The data that are managed within M Tree structure are represented by instances. The main goal of the structure is to provide information to students and course managers regarding the knowledge level reached by students. The proposed clustering procedure that is used for splitting full M Tree nodes is designed to properly classify learners. A baseline classification scheme based on k-means clustering and a custom M Tree clustering are presented. For comparison, there are considered classical characterization formulas.

Povzetek: Opisana je metoda za izboljšanje učenja na osnovi M-dreves.

1 Introduction

The ability to classify a student's performance is very important in internet-based educational environments. A very promising area to attain this objective is the use of special designed data structure. In fact, one of the most useful applications of modern algorithms in e-Learning is classification. E-students are students that follow courses within an e-Learning platform. There are different educational objectives for using classification, such as: to discover potential student groups with similar characteristics and reactions to a particular learning strategy, to improve a student's capacity of learning, to group students who are failure-driven and help them improve their skills, to identify learners with low motivation and find remedial actions to lower drop-out rates etc. In the followings we have applied a classification method using unique algorithms which have a common base (tree classification).

One of the greatest challenges in e-Learning area is to continuously improve existing systems. In order to overcome the challenge there are needed sound procedures whose task is to prove the challenger procedure creates a better system than existing one. The key issue regarding effectiveness of educational process is classification. The main goal of this paper is to obtain a better or at least acceptable classification scheme with less computation power.

Some possible outcomes of such analysis process are: predicting students' grades (to classify in three classes/clusters of low priority – week learners, medium priority – easy learners, high priority – competitive learners) from test scores.

Clustering algorithms are part of the unsupervised classification techniques. They try to group a set of items into subsets or clusters. The cluster algorithms' goal is to create clusters that are coherent internally, but clearly different from each other. In other words, items within a cluster should be as similar as possible; and items in one cluster should be as dissimilar as possible from items in other clusters. In this paper, learners represent items.

The standard k-means algorithm [1] is used as baseline unsupervised classifier. K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared Euclidean distance of items from their cluster centres where a cluster centre is defined as the mean or centroid of the items in a cluster.

M-tree [2,3] is a dynamic access method suitable to index generic "metric spaces", where the function used to compute the distance between any two objects satisfies the positivity, symmetry, and triangle inequality postulates. The M-tree design fulfils typical requirements of multimedia applications, where objects are indexed using complex features, and similarity queries can require application of time-consuming distance functions. In this paper we describe the basic search and management algorithms of M-tree, introduce several heuristic split policies, and experimentally evaluate them, considering both I/O and CPU costs. The obtained results also show that M-tree performs better than R*-tree on high-dimensional vector spaces.

2 Related Work

It is now recognized that e-learning further requires the means to summarize and classify learner trends and patterns. One serious candidate solution is DM (data mining), already quite successful in e-commerce and bio-informatics, where results are achieved through the use of associates, classifiers, clusters [8], pattern analysers, and statistical tools.

Educational Data Mining [7] is an emerging discipline concerned with developing methods for exploring the unique types of data that come from educational settings, and using those methods to better understand students, and the settings which they learn in.

Since the mid-1990's, e-learning has epitomized a broad range of learning categories while reinforcing four major pedagogical perspectives often neglected during e-learning system development. First, insight from cognitive learning processes can shed light on how the brain functions. Second, emotional aspects of learning can be traced, such as interest, motivation, interaction, fulfilment, and enjoyment. The third perspective incorporates skills and behaviours, such as role-playing, that are particularly useful in real settings.

Lastly, a social perspective involving the interaction with other people permits a focus on collaborative discovery, namely, the interplay of peer pressure and support. The complexity of the approach is high, and the specialists have been more preoccupied about the development of the information systems from the perspective of the technological informatics infrastructure. The studies devoted to the technology infrastructures embedded in the information systems are insufficiently presented in literature [5].

One of the constructive steps in this direction was done by V. Fomichov and O. Fomichova in [6]. The authors introduced the notation of conceptual-visual dynamic schemes (CVD-schemes). The CVD-schemes are the marked oriented graphs introduced in cognitonics domain for inventing effective teaching analogies. Such graphs establish a correspondence between the components of a piece of theoretical material to be studied and the components of a well-known or just created by the teacher but bright fragment of the inner world's picture of the learner.

Novel database applications, such as multimedia, data mining, e-commerce, and many others, make intensive use of similarity queries [2] in order to retrieve the objects that better fit a user request. Since the effectiveness of such queries improves when the user is allowed to personalize the similarity criterion according to which database objects are evaluated and ranked, the development of access methods being able to efficiently support user-defined similarity queries becomes a basic requirement. In this article we introduce the method called the M-tree that can process user-defined queries in generic metric spaces, that is, where the only information about indexed objects is their relative distances. The M-tree is a metric access method that can deal with several distinct distances at a time: (1) a query (user-defined) distance, (2) an index distance (used to build the tree), and (3) a comparison (approximate) distance (used to quickly discard from the search uninteresting parts of the tree). We develop an analytical cost model that accurately characterizes the performance of the M-tree and validate such model through extensive experimentation on real metric data sets. In particular, our analysis is able to predict the best evaluation strategy (i.e., which distances to use) under a variety of configurations, by properly taking into account relevant factors such as the distribution of distances, the cost of computing distances, and the actual index structure.

The access method called M-tree is proposed to organize and search large data sets from a generic "metric space", i.e. where object proximity is only defined by a distance function satisfying the positivity, symmetry, and triangle inequality postulates. The M-tree design has been motivated by retrieval requirements from typical multimedia database applications, where objects, such as text, image, and video, are indexed using complex feature representations, and search for objects similar to a query object can involve application of time-consuming distance functions. We detail algorithms for insertion of objects and split management which keep the M-tree always balanced - several heuristic split alternatives are considered and experimentally evaluated. Algorithms for similarity (range and k-nearest neighbours) queries are also described. The results from extensive experimentation with a prototype system are reported, considering as the performance criteria the number of page I/O's and the number of distance computations. The results demonstrate that the M-tree indeed extends the domain of applicability beyond the traditional vector spaces, performs reasonably well in high-dimensional data spaces, and scales well in case of growing files.

As our project goal is to bring a contribution to E-learning domain, our idea is to structure didactic chapters as concept maps and provide an efficient electronic students distribution by their results. The chapter's notions will be split depending on their level of priority as follows:

- Low priority notions – referring to introductive notions about the chapter.
- Medium priority notions – referring to basic notions of the chapter.
- Maximum priority notions – referring to advanced notions of the chapter.

As our concept maps are represented as tree structures, where each path of the tree is assigned a weight, these priorities can be computed depending on the assigned weights.

Assuming that, in order to evaluate a number of students, each chapter presents a final quiz, we have decided to use concept maps as weighted trees in order to generate tests containing notions of different priority levels. The algorithms responsible for generating tests are based on tree searches methods.

3 Building Clusters with M Tree and k-Means Algorithms

Our implementation uses the following student's data structures:

```
struct Student{
    int IDStudent;
    float Lp;
    float Mp;
    float Maxp;
    int Where[3];
};
```

The *IDStudent* is the identifier corresponding to each student entering the online distribution program. In order to evaluate these students, each chapter presents a final quiz containing notions belonging to the previously discussed levels of priority. Thereby , each student will

get a mark for each type of notion, contained in the chapter:

- a *Lp* mark corresponding to *low priority notions,*
- a *Mp* mark corresponding to *medium priority notions,*
- *MaxP* mar corresponding to *maximum priority notions*.

Each one of these marks is important , because they represent the guiding tool for a student. Example:

Let us suppose that the student identified by his/her *IDStudent=1002* takes the quiz, at the end of the chapter and gets the following results: (Lp= 7.70, Mp=6.78, Maxp=5.00).

Right know, a teacher, or even an electronic program, is able to compute the minimum performance of this student, reaching the conclusion that the advanced notions of the chapter(indicated by Maxp) have not been covered properly by this student, and therefore , he/she needs to put more energy in this direction. These directions are given by *Where* vector, used for providing instructions regarding where should a student improve his/her level of knowledge. A graphical representation is given below:

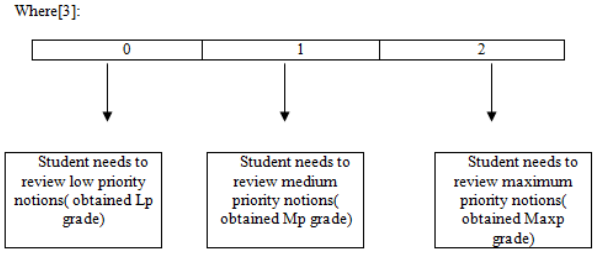

Figure 1: *Where* vector, used for providing instructions regarding where should a student improve his/her level of knowledge.

As a start, Where[3]=(0,0,0). If any of this vector's component changes to 1 , this means it becomes active. For instance, if StudentA.Where[3]=(0,1,0), it means that he/she needs to review notions of medium priority level.

These students are then, distributed and placed by our algorithm in a M-tree structure. Before moving on with the algorithm, let us present the structure of the M-tree. As it was explained above, the M-tree is a spatial, metric tree, consisting of: *1 root* and *k leaves* containing students.(As we will see later, these leaves represent clusters of students). For now, let us stick to the structure of a M-tree, mentioning that this tree is actually a spatial one, where its leaves can be imagined as spheres, containing points, which are actually students. As far as the structure of a node is concerned, we have:

The M-Tree node's structure is:

```
struct m_Node{
    int nrKeys;
    bool isLeaf;
    float radius [NMAX];
    m_Node *routes [NMAX];
    struct Student students[NMAX];
};
```

The `nrKeys` represents the number of students contained in a node (cluster). As far as our M-tree is concerned, we pay extra attetion to the nodes, because it is very important wheather they are leaves(terminal nodes) or internal nodes, as we will see later in the algorithm. The boolean variable `isLeaf` is pointing out our exact concern: weather a node is leaf or internal. Moving on, as itwas previously said, we consider our nodes as spheres, and as any sphere, it is geometrically represented by its center *C(x,y,z)* and its radius *R*. However, in order to match these notions to our real implementation, we have constructed an abstract interpretation for this geometrical representation. The center *C* of a sphere(cluster) will be represented by the average student belonging to the set of students contained in that particular cluster. Instead of spatial coordinates *(x,y,z)* , our *centerStudent* will be represented by its elements *(IDStudent,Lp,Mp,Maxp)*, which were presented earlier. The *radius* of this abstract sphere will be represented by the distance between the *centerStudent* and the student (students) with the lowest results in that cluster. We will take a closer look to this abstract system later, when we will discuss the implemented algorithm. The *routes* represent the children of a particular node and of course, the `nrKeys` points inside the sphere, which are actually the *students*, as in our abstract system, a spatial point is represented by a student.

Our implementation is based on the idea of students distribution depending on their results to a quiz at the end of a chapter. For a better understanding of our implementatio let us consider a real situtation:

Let us consider a finite set S of k students defined as $S=\{St_1,St_2,....St_k)$, $k>0$. Let us supose that all these students have taken a quiz at the end of a chapter in order to evaluate their level of knowledge. Each student is represented by his/her IDStudent, and his/her grades: Lp, Mp, Maxp (they were discussed earlier). Let us assume that we want to create an hierarchy among these students, depending on their results. In order to do that, we need to group these students in *clusters*, each cluster having its own attribute. An attribute, for a cluster, represents the level of performance for that particular group of students. Moreover, these attributes are also used as indicators pointing out to the type of notion (low priority, medium priority, maximum priority) the student needs to review. After a group of students (cluster) is formed, a center is chosen, that is the average student in that group, and all the other students are distributed in a spherical manner, arround him.

Computation of a radius for a cluster : the radius of a cluster represents the maximum possible distance between the *centerStudent* and the rest. The bigger the distance is, the better or the lower the results of that student are. Just as in real cases, when we say there is a big distance between this *average student* and *student A*, this means that Student A has either better results or worse results, we don't know for sure. Anyhow, should the distance between the *centerStudent* and any other student be greater than the cluster's radius, it means that the particular student does not belong to that cluster, for

the simple reason that he/she is smarter than all those students in that cluster or his/her results are lower than any other's in that cluster.

The radius of a cluster is computed, depending on the results of each student, being the maximum distance between two students. We define the distance between *StudentA and StudentB* as follows:

$$d_{AB}= \max\{(|Lp_A-Lp_B|, |Mp_A-Mp_B|, |Maxp_A-Maxp_B|)\} \quad (1)$$

As you can see, when we measure the distance between two students, we are looking for the most marking difference between them. This also helps us in defining attributes of a cluster, depending on the type of notion students should focus on (low priority notions, medium priority notions, maximum priority notions). Moreover the relation (1) guarantees that the radius of a cluster represents the biggest difference between the levels of knowledge for each student belonging to that cluster.

As example, let us consider the student A with his/her results: (9.60, 8, and 7.50). Let us consider the student B with his/her results: (7.60, 8, 6.50). Following the relation (1), we get the biggest difference 2 (9.60-7.60). This is the biggest distance between them two. So they might have similar knowledge for medium priority notions (8,8), and maximum priority notions(7.50,6.50), but when it comes to low priority notions, we see a gap between them(StudentA -9.60 , StudentB -7.60). Let us consider that StudentB has the lowest result in the cluster, and StudentA is the centerStudent. Then, as we have presented earlier , they can be grouped in a cluster with its radix 2 . Let us suppose now that StudentC gets the results: (5, 6.30, 5). We get:

$d_{AC}= \max \{|9.60-5|,|8-6.30|,| 7.50-5|\}=4.60,$

$d_{BC}= \max \{|7.60-5|,|8-6.30|,| 6.50-5|\}=2.60.$

As you can see, neighther of these distances is lower than our cluster's supposed radius, as the difference between Student and the students StudentA, StudentB is huge, so there is no way, StudentC will not become a member of this cluster. Moreover, based on the present results of StudentA and StudentB , we can define an attribute for this cluster: all students belonging to this cluster, will posses similar knowledge levels for medium and maximum priority notions, but the marking difference between them will be represented by the low priority notions, so all of them need to review this part of the chapter.

The main steps of the agorithm are:

Step1. We start from a simple representation of students, identified by their elements:

- IDStudent
- Lp (score)
- Mp (score)
- Maxp (score)

Step2. We picture the set of the students who have taken the test as the points in 3D space. Our algorithm involves two major operations:

- a clustering operation
- a split operation

We will first describe the *spliting* method. We have decided that these groups of students should have a maximum number of allowed members. Let us denote this number as the *filling factor* of a cluster (student group). Whenever the number of students in a particular cluster becomes greater that this filling factor, a cluster splitting is involved. This is how the M-tree extends its nodes. The splitting procedure works as follows:

At the beginning two random students from that cluster are chosen as the centers for the new clusters resulting after splitting. Let us denote them *Student1* and *Student2*. Next, we distribute the rest of the students arround the new centers *Student1* and *Student2*. If, for instance, we have *Student1* and *Student2* as centers, the question is where should we attach *Student3* to? We compute the distance between (Student1, Student3) and *(Student2, Student3)*, using relation (1). *Student3* will go near that student which is more closed to him/her (that is, from a level of knowledge point of view). After the distribution is completed, we start the *chooseCenter* method, which recalculates the new centers of the clusters, and if new centers are found, the entire discussed process happens again, until new centers are found no more. This process is called the *Clustering Process*. After that the effective splitting happens, and the initial tree node is split into two nodes. When these clusters are formed, they are also assigned atributes(we have discussed about them earlier). What is interesting is that these atrributes suffer a constant evolution, depending of the students inserted in that specific cluster. As the clusters suffer constant splittings and modifications, whenever the number of students inside is greater than the filling factor, so do the attributes change, transforming a part of the old cluster in a better one (shelters students with higher levels of knowledge) or even a lower one (shelters students with lower levels of knowledge).

The classical M Tree algorithm has been adapted such that the final structure has two levels. The procedure for building the structure takes into consideration both the desired number of clusters and the filling factor of a leaf node.

```
procedure MTree (x1, x2, …, xN; K; F)
// K – the number of clusters
// F– filling factor
 for ( i=1, i<N ){
   Ci = FindCentroid (centroids, xi);
   if ( #Leaf [Ci] has F instances)
     if (#we have k clusters)
       #put xi in Leaf [Ci]
     else
       #split Leaf [Ci]
   else
     #put xi in Leaf[Ci]
   RecomputeCentroids(Leaf[Ci])
 }//end for
```

The computational complexity of this M Tree procedure takes into consideration that each instance is considered only once. That is why the complexity of this operation is *O(N)*. Still, the number of clusters influences the complexity since the best corresponding cluster needs to be determined. The time taken for this operation is *O(K)*. Still, the recomputation of the centroid is not so

costly as in the case of k-means algorithm due to the filling factor parameter. Thus, the overall complexity of employed M Tree procedure is O(KNF).

Clustering and splitting procedures in pseudocode are presented below.

```
Procedure Cluster_and_split
 #Get the cluster which will suffer splitting procedure;
 #Let StudentSet={Si| Si ϵ cluster, i=0, nrKeys, Si student};
 #Choose
        S1 ϵ StudentSet (as new center);
        S2 ϵ StudentSet (as new center2);
        oldCenterStudent1←S1;
        oldCenterStudent2←S2;
 #While (!STOP_CONDITION) do {
        For each other Si ϵ StudentSet, (Si ≠ S1 and Si≠ S2 ) do {
     Compute d1i=max{(|Lp1-Lpi|,|Mp1-Mpi|,|Maxp1-Maxpi|);
     Compute d2i=max{(|Lp2-Lpi|,|Mp2-Mpi|,|Maxp2-Maxpi|);
     If (d1i < d2i)
        Attach Si to the cluster with center S 1
     else
        Attach Si to the cluster with center S 2
     #Determine attributes
    }//end do

newCenterStudent1=chooseCenter(newFormedCluster1, newClusterRadix1);

newCenterStudent2=chooseCenter(newFormedCluster2, newClusterRadius2);
  STOP_CONDITION←
        (newCenterStudent1=oldCenterStudent1)
        && (newCenterStudent1=oldCenterStudent1))
  #Effective_split_of_initial_cluster
  }//end while
```

The *Determine attributes* sequence works in the following way. Every time a distance between two students is computed, after the marking difference between them is extracted, inside the *Where[3]* vector, the corresponding component of that type of level notion the maximum was extracted for, is incremented (we say a flag is raised for that component).

Example. Let us consider that the marjing difference between studentA and studentB regards the low priority notions, then studentA.Where[0]=1 and studentB.Where[0]=1 (see the *Where[3]* vector configuration in previous sections). Notice that this *Where[3]* vector changes for every new student in the cluster, because in the end of the clustering process only the attributes of the average student will prevail, because he/she is the center of that particular cluster.

The *chooseCenter* sequence simply computes the biggest distance between all the students and also sets the new studentCenter of the cluster, as we will see in the complete algorithm's pseudocode.

Now we will take a closer look to the *splitting procedure*:

```
Procedure Effective_split_of_initial_cluster
    m_Node = node for splitting
 m_node→isLeaf = false;
 //the initial node becomes a root,
 //where the centers of the clusters will be retained
 # Alloc memory for leftChild and rightChild of m_node;
 m_Node→leftChild = newCluster1;
 m_Node→rightChild = newCluster2;
 #Insert S = { centerStudent1|radiusCluster1,
        centerStudent2|radiusCluster2} in m_Node
 #Attach the rest of students Si
        to leftChild or rightChild
 leftChild→isLeaf=true;
 rightChild→isLeaf=true;
```

After seeing these two important steps in the algortihm, it is time to present the entire process:

```
Procedure Build_M_Tree
   #initializeTree;//create an empty root tree
                          //and set root as leaf
   #While (not endOfFile){
    #Get input data student(IDStudent, Lp,Mp,Maxp);
    #If  ( tree->root->nrKeys < filling factor )
     #make a simple insertion in the actual root
    #Else
     #apply Cluster_and_split;
   }//end while
```

The classical standard k-means algorithm partitions a dataset on N instances into K clusters. The algorithm is:

```
procedure k-means (x1, x2, …, xN; K)
 {c1, c2, …, cK} ← Select Random Centroids
 for ( k=1, k<K )
  centroidk = ck;//these are initial centroids
 while (#centroids are not same){
  for ( k=1, k<K ){
   for ( n=1, n<N ){
    j = index of corresponding cluster
    #put xn in corresponding cluster Cj
   }//end for
  }//end for
  for (k=1, k<K)
   # compute centroids for all clusters
 }//end while
```

The most important discussion regards the computational complexity of k-means algorithms. Most of the time is spent on computing distances between items. This computing is performed when putting instance x_n in cluster C_j. One such operation costs *log(M)*. The reassignment step computes *KN* distances, so its overall complexity is *log(KNM)*. In the re-computation step, each vector gets added to a centroid once, so the complexity of this step is *log(NM)*. For a fixed number of iterations I, the overall complexity is

therefore *log(IKNM)*.Thus, K-means is linear in all relevant factors: iterations, number of clusters, number of vectors, and dimensionality of the space. One of the most important issues regards the number of iterations. In most cases, K-means quickly reaches either complete convergence or a clustering that is close to convergence. In the latter case, a few items would switch membership if further iterations are computed. This computation has a small effect on the overall quality of the clustering.

4 Clustering Evaluation Metrics

A comparison of two distinct procedures defines the needed steps in order to obtain sound results. The presented analysis procedure compares two methods that are used for building clusters of items: k-means and M Tree.

An input dataset is considered. Both k-means and M Tree algorithms are than used for building clusters from the same dataset. For each set of clusters, there are computed specific indicators for characterizing obtained clusters. The indicators that are taken into considerations are:

Tightness Indicator:

$$Q = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{x \epsilon C_i} d(x, \mu_i)$$

where $|C_i|$ is the number of points from cluster i. The value for Q will be small if the data points from the cluster are close. Thus, in the comparison analysis procedure the clusters with smaller computed values of Q have higher quality.

Homogeneity Indicator:

If the centroids of clusters are computed with formula $r_k = \frac{1}{n_k} \sum_{x \epsilon C_k} x$, where x are the instances from cluster C_k , than homogeneity indicator is

$$H(C) = \sum_{k=1}^{K} \sum_{x \epsilon C_k} d(x, r_k)^2$$

The value for H will be small if a cluster has homogeneous structure. This, in the comparison analysis procedure the clusters with smaller computed values of H have higher quality.

Cluster Distance:

$$CD = \sum_{1 \le j \le k \le K} d(r_j, r_k)^2$$

where j and k are indexes of clusters whose centroids r are taken into consideration. The value for CD will be big if the similarity among clusters themselves is low. Thus, in the comparison analysis procedure the methods with bigger computed values of CD have higher quality.

Weakest Link between Points:

The weakest link for a cluster is the maximal value of all pairs of points belonging to the same cluster.

$$WL = \max (d(x_i, x_j)),$$

for all x_i and x_j belonging to the same cluster.

5 Experimental Results

The first experiment builds an M Tree from the data from 6 students. The values for (Lp, Mp, Maxp) = {(8,7,6), (5,9,10), (7,7,7), (7,8,7), (6,6,6), (4,3,5)}. The value of filling factor is 4.

We insert students with IDStudent 1,2,3,4 in one cluster. Student 5 also needs distribution, but adding him/her to the same cluster is not possible, since a cluster can hold a maximum of 4 persons (filling factor is 4). So a split is mandatory.

After split the clusters are:

Cluster1={Student1, Student3, Student4},

Cluster2={Student2}.

Then, student 5 may be inserted, and the clusters become as follows:

Cluster1={Student1, Student3, Student4, Student5} ,

Cluster2={Student2}.

A new split is necessary, because cluster 1 is full. Thus, the following clusters are being obtained:

Cluster1={Student3, Student4, Student5}

Cluster2={Student2}

Cluster3={Student1}

Finaly, Sudent6 is inserted in cluster 3, and thus the following clusters are obtained:

Cluster1={Student3, Student4, Student5}

Cluster2={Student2}

Cluster3={Student1, Student6}.

The *chooseCenter* sequence simply computes the biggest distance between all the students, also it sets the new studentCenter of the cluster, as we will see in the complete algorithm's pseudocode.

The second experiment takes into consideration 15 students. For each student there are available two weighted parameters: the number loggings and the time spent on-line. The real parameters scale such that all values are in the range 0 to 16.

The input dataset is:

A1(10.94 , 11.86); A6(11.02,2.28); A11(9.29 , 13.86); A2(1.58 , 6.27); A7(11.23,9.37); A12(8.00 , 1.09); A3(13.66 , 4.62);

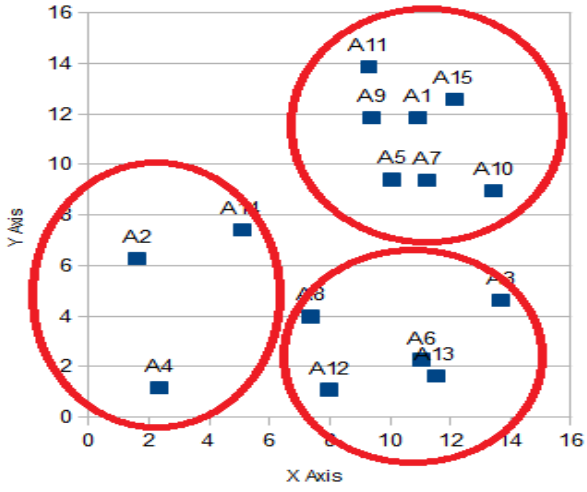

Figure 2: Distribution of learners with k-Means.

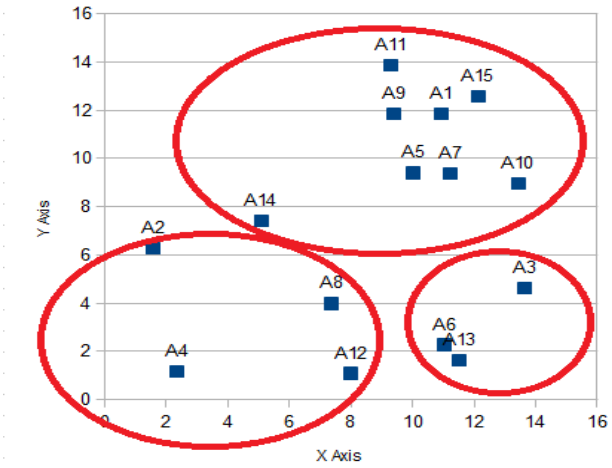

Figure 3: Distribution of the learners with MTree.

A8(7.35 , 3.99); A13(11.52 , 1.63); A4(2.33 , 1.16); A9(9.4 , 11.84); A14(5.08 , 7.42); A5(10.04 , 9.41); A10(13.43 , 8.97); A15(12.12 , 12.59);

This dataset is used for building three clusters with both k-means and M Tree algorithms. The obtained distributions are presented in figures 2 and 3.

The obtained clusters by k-Means clustering have the following centroids and composition:

C1 (2.99,4.95) *//Cluster 1's Centroid*
A2 (1.58, 6.27)
A4 (2.33, 1.16)
A14 (5.08, 7.42)

C2 (10.92, 11.13) *//Cluster 2's Centroid*
A1 (10.94, 11.86)
A2 (10.04, 9.41)
A7 (11.23, 9.37)
A9 (9.40, 11.84)
A10 (13.43, 8.97)
A11 (9.29, 13.86)
A15 (12.12, 12.59)

C3 (10.31, 2.72) – Cluster 3's Centroid
A3 (13.66, 4.62)
A6 (11.02, 2.28)
A8 (7.35, 3.99)
A12 (8.00, 1.09)
A13 (11.52, 1.63)

The obtained clusters by M Tree clustering have the following centroids and composition:

C1 (10.04, 9.41) - *//Cluster 1's Centroid*
A1 (10.94, 11.86)
A2 (10.04, 9.41)
A7 (11.23, 9.37)
A9 (9.40, 11.84)
A10 (13.43, 8.97)
A11 (9.29, 13.86)
A14 (5.08, 7.42)
A15 (12.12, 12.59)

C2 (2.33, 1.16) *//Cluster 2's Centroid*
A2 (1.58, 6.27)
A4 (2.33, 1.16)
A8 (7.35, 3.99)
A12 (8.00, 1.09)

C3 (11.02, 2.28) *//Cluster 3's Centroid*
A3 (13.66, 4.62)
A6 (11.02, 2.28)
A13 (11.52, 1.63)

For each clustering procedure there were computed the evaluation metrics presented in section 3. The results are presented in the following table:

Indicator	Clustering Procedure	
	k-means	M Tree
Tightness	7.55	8.52
Homogeneity	100.47	137.48
Clusters Distance	230.47	203.11

Table 1: Tightness, homogeneity and cluster distance indicators for k-means and MTree distributions.

The link analysis for both distributions is presented in the following table:

Indicator	Clustering Procedure	
	k-means	M Tree
Weakest Link Cluster 1	0.9	1.21
Weakest Link Cluster 2	0.84	1.15
Weakest Link Cluster 3	0.87	0.51

Table 2: Weakest link values obtained for k-means and MTree distributions.

The k-means results are obtained using Weka [4]. Weka is a collection of machine learning algorithms for data mining tasks which has implemented the k-means clustering algorithm.

The M Tree results are obtained, using a custom Java implementation of the algorithm. The main differences of this implementation compared with classical M Tree algorithm regard two aspects. One regards the general structure of the tree that is restricted to two levels. This means there is only one root node where centroids along with covered radius are placed. The second issue regards the way *k* (the number of clusters) and f (the filling factor) are managed. If the algorithm is required to produce a certain number of clusters, the instances are placed into appropriate clusters until a filling factor is reached. When this happens, a split is performed. Splitting is no longer performed when the desired number of clusters is reached. In this way, the clustering process is directly managed by the values *k* and *s*.

The comparison of the two obtained distributions reveals the fact that the M Tree distribution clusters have lower quality than the ones obtained by usage of k-means. Still, the results obtained by M Tree are very different from the ones obtained by k-means. All indicators presented in table 1 have better results for k-means than the ones obtained for M Tree. It can be observed that the tightness and homogeneity are better (because they have smaller values) for k-means than for M Tree.

Another comparison that may be done regards the mobility of centroids. Although the differences of computed indicator (Tightness, Homogeneity, Clusters Distance) values are not very small, the computed centroids are quite close.

6 Conclusions

The paper presents a study usage of an implementation of M-Trees building algorithm. The tree manages real data representing e-Students (students from an e-Learning environment). The instances (i.e. the students) are characterized by attributes representing the scores obtained when taking tests. The tests are classified according with their level of difficulty: low, medium and high.

Within classical M-Tree building procedure it was used a custom clustering procedure when splitting a node was necessary. The clustering procedure is designed such that produces an optimal grouping of students regarding the "distances" in knowledge among them.

The tests were performed with datasets representing 200 students, and the filling factor of a cluster was restricted to 18. As a result, we got 7 clusters, with attributes, leading e-Students to notion reviews.

Another goal of our current implementation is to provide valid data distribution using specific data validation algorithms. Moreover our algorithm is able, for the moment, to distribute e-Students which test their level of knowledge for one chapter only. We wish to extend this process for multiple chapters.

This paper also presents a procedure that measures the degree in which the effectiveness of an e-learning process has improved. The analysis process is data centred. The data represents experiences provided by learners. In this study two features (attributes) characterize each learner: the number loggings and the time spent on-line.

The goal of the procedure is to produce clusters of users using two different techniques: standard k-means algorithm implemented in weka and a custom flavour of M Tree algorithm with a custom implementation.

The input dataset is restricted to a sample of 15 learners. This choice is because a manual inspection of the obtained clusters is desired. An automated analysis of the obtained clusters is performed by computing some basic clustering quality metrics: Tightness, Homogeneity, Clusters Distance and link analysis. The obtained results show an acceptable quality of the M Tree clusters although the computational complexity of the algorithm is much lower than complexity of k-means.

The main goal of the paper is to find an algorithm that produces acceptable results with complexity much smaller than a classical procedure.

The quality of the obtained clusters has a direct influence over the degree in which the e-learning process has been performed. Unsupervised classification (clustering) is one of the main methods for making evidence regarding the knowledge acquisition of learners. Once a high quality distribution has been discovered, a learner may by clustered at certain moments and progress may be evaluated. Of course, the process needs to be well defined and needs to be based on a high quality clustering procedure.

The future works regard different aspects. A first issue would be to replicate the procedure with more data. This may be accomplished on hundreds or even thousands of learners, if data are available. The clustering procedure is highly influenced by the initial centroids. In custom initialization is advisable. A good starting point may be obtained by using a k-means clustering on a sample dataset from the entire dataset. The quality of the clustering process is directly influenced by the choices made regarding k and f values. Thus, an initialization step may also refer to prior computation of the optimal number of clusters and optimal filling factor. The computation of these parameters may be delegated to other high quality clustering procedure that works on a data sample.

Finally, there may be defined procedures for assessing progress in time and even recommendations. The progress in time may be computed classifying the learner from time to time. This may yield to a learning path that has been followed by the learner. More than this, there may be obtained recommendations for the learner. The recommendations may regard necessary actions necessary to be taken by the learner in order to improve his learning curve.

References

[1] J. Hartigan and M. Wong (1979). A k-means clustering algorithm. In *Applied Statistics*, 28, pp. 100–108.

[2] Uhlmann, J.K. (1991). Satisfying General Proximity/Similarity Queries with Metric Trees. *Information Processing Letters*, Vol. 40, pp. 175-179.

[3] Ciaccia, P., Patella, M., Zezula, P. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*. Morgan Kaufmann, pp. 426-435.

[4] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009). The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Vol. 11, Issue 1.

[5] Burlea Schiopoiu A. (2008). The Complexity of an e-Learning System: A Paradigm for the Human Factor, *The Inter-Networked World: ISD Theory, Practice and Education*. Vol. 2, Springer-Verlag, New York, pp. 267-278.

[6] V. Fomichov and O. Fomichova (2006. . Cognitonics as a New Science and Its Significance for Informatics and Information Society. Special Issue on Developing Creativity and Broad Mental Outlook in the Information Society (Guest Editor Vladimir Fomichov), *Informatica. (Slovenia),* 2006, 30 (4), pp. 387-398.

[7] C. Romero and S. Ventura (2007). Educational Data Mining: A Survey from 1995 to 2005. *Expert Systems with Applications*, 33(1), pp. 135-146.

[8] Gema Bello Orgaz, Héctor D. Menéndez, David Camacho (2011). Using the Clustering Coefficient to Guide a Genetic-Based Communities Finding Algorithm, *IDEAL 2011,* pp.160-169.