Proceedings of Engineering & Technology (PET)
pp. 610-616

An improved multi-database classification approach

Salim Miloudi, Sid Ahmed Rahal Hebri, Salim Khiat
Department of computer science
University of Sciences and Technology of Oran Mohamed Boudiaf (USTO-MB)
BP 1505 El M'Naouar, 31000 Oran, Algeria
salim.miloudi@univ-usto.dz, Rahalsa2001@yahoo.fr, salim.khiat@univ-usto.dz

***Abstract*— For business decision making purpose, multi-branch companies may need to mine their databases distributed through their branches in order to extract useful information called patterns. Traditional data mining techniques integrate all the data from these databases to amass a huge dataset for pattern discovery. However, this approach may generate an expensive search cost for centralized processing and might disguise some important patterns. To deal with the latter problems, we may need to classify the multiple databases into clusters of similar databases. Each cluster could be analyzed individually to discover new patterns reflecting specific information about each group of branches. Inspired by the existing works, this paper presents an improved multi-database classification (MDC) approach which enhances the quality of the patterns mined from the databases and optimizes the time complexity of the classification algorithm.**

Keywords*— *Multi-database classification (MDC); Database clustering; Database selection; Relevance factor;Similarity measure; Goodness measure; Pattern Analysis; Frequent-itemsets; Association rules

I. INTRODUCTION

Large organizations have different types of business distributed through their branches and each branch may have its own database which collects continuously transactions from customers. For analyzing purpose, it is important to mine these multiple databases to discover new patterns (e.i., frequent itemsets and association rules) useful for the decision making process. Traditional data mining techniques known as *mono-database mining* [1-3] integrate all the data from these databases to form one dataset for knowledge discovery. Nevertheless, this process may accumulate a huge database for centralized processing and may hide some important patterns reflecting the characteristics of the branches. For example, a pattern such as "75% of the branches in certain regions observe that 30% of coffee's purchases imply also the purchase of sugar" cannot be discovered by using the mono-database mining. In addition, it's not always obvious to integrate all the data from the multiple databases due to the data privacy and data irrelevancy issues. To overcome the latter problems, we need to identify the groups of databases that enhance the quality of the patterns discovered when they are mined together. This approach is referred to as the *multi-database classification* (MDC), which aims to classify the multiple databases into disjoint clusters sharing common features. For example, if a large company has 10 branches including 5 branches for household appliances, 3 branches for clothing and 2 branches for food, we cannot apply existing data mining techniques over the union of the 10 branch databases. We might lose the data distribution. That's why, we need to classify them into 3 clusters according to their type of business, and then we could analyze each database cluster individually. Consequently, mining the multiple databases becomes more manageable. Few MDC algorithms have been proposed in the literature [4-7]. They differ in (1) the similarity measure used to identify the relevant databases, (2) the goodness measure used to assess the classification and (3) the execution time taken to find the best classification. The accuracy of a MDC algorithm is highly determined by the similarity measure used to calculate the relevance between databases. In the work proposed by A.Adhikari and PR .Rao[9], more the number of frequent itemsets shared between databases is large, more cohesive are the clusters discovered. However, existing algorithms in [4-7] neglect some infrequent itemsets, which are not supported by a certain databases (e.i, their support values are less than the user-defined threshold), but these itemsets may become frequent when the databases are merged and a traditional data mining technique is applied. In this paper, we propose an improved similarity measure based on maximizing the number of frequent itemset discovered by the multi-database mining process [1-3]. Thus, only relevant databases that generate more frequent itemsets are put into the same cluster. To do so, we use the synthesizing model proposed in [8] combined with the correction factor defined in [9] to estimate the support of a frequent itemset in the union of two databases. In the other hand, we improve the time complexity of the classification algorithm by proposing a graph-based classification approach which represents the similar databases as a set of vertices each pair of which is connected by an edge. The experiments carried on public databases show the efficiency of our approach against the existing works.

The rest of the paper is organized as follows. Section II presents in details the existing MDC approaches proposed in the literature. Section III discusses and compares the different MDC algorithms. In Section IV, we present our contribution to improve the existing works and Section V concludes the discussion and highlights the future works.

ISSN: 2356-5608

II. Presentation of the MDC Approaches

In the literature, we distinguish two categories of MDC approaches according to whether they are dependent or independent application. In this paper, we focus on studying the independent application approaches.

A. Dependent-application Approach for MDC

A dependent-application strategy for MDC, referred to as *database-selection* [10], aims to identify databases which are relevant to a certain user-request denoted *Q*. The authors H. Liu, H. Lu and J. Yao [10] have proposed a relevance factor *RF*, to measure the deviation between *Q* and the attributes of a relational table. The time complexity of the proposed algorithm is $O(m\times N\times M)$ times, such that N and M are the maximum number of records and attributes in the *m* databases respectively. Obviously, for large databases, the time complexity is going to increase significantly.

Database selection represents a feasible way to avoid joining irrelevant databases with the relevant ones and then it reduces the search space to explore and enhances the quality of the patterns discovered with the reference to the user-request. However, for real world applications, this method cannot be applied when the multiple databases are mined without specifying a user application. Moreover, due to the privacy issue, some branches may refuse to share their original row data. In this case, database selection isn't appropriate since it requires a direct access to the records of each database. The problems above motivated many authors to design a new MDC algorithms independent-application [4-7] to deal with the limitations of the dependent-application approach.

B. Independent-application Approach for MDC

Given a set of *n* databases $D=\{D_1,D_2,\ldots,D_n\}$ corresponding to the *n* branches of an inter-state company, an independent-application classification strategy is referred to as database clustering, which aims to search for the best classification of the *n* databases without specifying any application. To do so, existing MDC algorithms proposed in [4-7] generate $m\in\left[1,\frac{n^2-n}{2}\right]$ candidate classifications incrementally until finding the best classification which optimizes a certain *goodness* measure. In general, an MDC algorithm performs as follows:

a) Build a similarity matrix of size $n\times n$ using a similarity measure *sim*. In Section III, we present some similarity measures.

b) For each distinct similarity value δ listed in a sorted order, a candidate classification $class(D,sim,\delta)$ is generated such that two database D_i and D_j are put in the same cluster if $sim(D_i,D_j) \geq \delta$. In [4], each similarity level δ is selected based on a step value λ given initially by the user.

c) Evaluate each candidate classification using a quality measure *goodness.*

In the following section, we present a comparative study of the different MDC algorithms proposed in the literature.

III. Discussing and Comparing Existing MDC Algorithms

The authors Wu X, Zhang C and Zhang S in [4] were the first to propose a new approach independent-application for classifying transactional databases. Thus, two similarity measures, sim_1 and sim_2 have been proposed to group similar databases into disjoint clusters without specifying any application.

1) Definition: The similarity $sim_1(D_i,D_j)$ is computed based on the set of items shared between two transactional databases D_i and D_j and it is defined as follows.

$$sim_1(D_i,D_j) = \left|Items(D_i)\cap Items(D_j)\right| \Big/ \left|Items(D_i)\cup Items(D_j)\right| \quad (1)$$

Where $|Items(D_i)\cap Items(D_j)|$ is the number of elements in the set $Items(D_i)\cap Items(D_j)$.

2) Definition: The similarity $sim_2(D_i,D_j)$ is calculated based on the set of items shared between the association rule sets S_i and S_j extracted from D_i and D_j respectively.

$$sim_2(D_i,D_j) = \left|Items(S_i)\cap Items(S_j)\right| \Big/ \left|Items(S_i)\cup Items(S_j)\right| \quad (2)$$

The number of items involved in computing sim_1 is more than that of sim_2. At a given minimum support and confidence threshold values, an algorithm for association rule mining may not extract any rule from a database. Hence, the accuracy of sim_1 is higher than that of sim_2.

Based on the previous similarity measures, an algorithm, *BestClassification* [4] has been proposed to search for the best classification of a databases set $D=\{D_1,D_2,\ldots,D_n\}$ minimizing a certain *goodness* measure. The algorithm calls a procedure *Greedy Class* to generate a *complete* classification, $class(D,sim,\delta)$, for each similarity level δ (initially defined by the user).

3) Defintion: Let $class(D,sim,\delta)=\{class^{\delta}_1, class^{\delta}_2,\ldots, class^{\delta}_k\}$ be a classification of *k* clusters generated at a similarity level δ. $class(D,sim,\delta)$ is *complete* if the following properties are verified.

a) $class^{\delta}_1 \cup class^{\delta}_2 \cup \ldots \cup class^{\delta}_k = D$

b) For any two clusters $class^{\delta}_l$ and $class^{\delta}_h$, $l\neq h$, $class^{\delta}_l \cap class^{\delta}_h = \emptyset$

c) $\forall D_i$ and D_i in $class^{\delta}_l$, $sim(D_i,D_j)\geq\delta$

d) $\forall D_i \in class^{\delta}_l$ and $D_i \in class^{\delta}_h$, $l\neq h$, $sim(D_i,D_j) < \delta$

Different classifications could be obtained by varying the similarity level δ. That's why Wu X, Zhang C and Zhang S[4] have proposed a goodness measure to evaluate each candidate classification in order to select the best one.

*4) Definition:*The goodness value of $class(D,sim,\delta)$ is defined as follows.

$$goodness(class,\delta) = \sum_{l=1}^{k} \sum_{D_i,D_j\in class^{\delta}_l}^{i\neq j} (1 - sim(D_i,D_j)) \quad (3)$$

goodness describes the sum of distances (1–*sim*(D_i,D_j)) between each database pair (D_i,D_j) in each cluster. Smaller the value of *goodness*, better is the classification.

*5) Definition:*To select the best complete classification, a distance measure called *distance*(*class*, δ) is defined as follows.

$$distance(class,\delta) = \left| goodness(class,\delta) - k \right| \quad (4)$$

Where *k* is the number of clusters in *class*(*D*,*sim*,δ). The complete classification which gets the smallest value of *distance*(*class*, δ) is selected as the best complete classification under the similarity level δ.

The time complexity of the proposed algorithm is $O(h \times n^4)$, such that *n* is the database number and *h* is the number of candidate classification generated before obtaining the best classification. Although good experimental results are obtained in [4] for certain similarity values, the time complexity of the algorithm remains high and becomes severe when the database number increases. Moreover, the proposed algorithm fails to find the best classification when the step size of searching λ is incorrectly initialized as shown in [5]. Also, in some cases, the while-loop of the algorithm doesn't terminate and may generate an infinite loop without finding any classification. The latter problem is due to the strong dependence of the algorithm on the similarity step size which is a user-input.

Motivated by the previous works, Li H, Hu X and Zhang Y [5] have modified the algorithm *BestClassification*[4] in order to optimize its time complexity and obtain correctly the best complete classification of a set of *n* databases. Thus, the same concepts (similarity and goodness measures) defined in [4] have been used. In order to avoid missing the best classification in case in which an incorrect step size has been defined, the distinct similarity values between the *n* databases are used as similarity levels to generate classifications. Thus, for each distinct similarity value sorted in the increasing order, a classification is produced. The proposed algorithm is optimal when comparing to *BestClassification*[4]. Its time complexity is $O(h \times n^3)$, such that *n* is the database number and $h \in \left[1, \frac{n^2-n}{2}\right]$ is the number of classification produced before obtaining the best classification.

The two algorithms above use the same similarity measure sim_1 [4] to cluster the multiple databases. Using a similarity measure based on items might be useful to estimate the correlation between large databases. In fact, extracting more information such as frequent itemset and association rules could be time consuming. However, sim_1 produces low accuracy in finding the correct similarity between two databases. The reason is that two transactional databases having many items in common are not necessarily similar.

In order to improve the accuracy of sim_1 [4], A.Adhikari and PR .Rao [6] have proposed a novel algorithm for multi-database clustering using a more accurate similarity measure, sim_3, based on the support of frequent itemsets shared between databases.

6) Definition : sim_3 is defined as follows.

$$sim_3(D_i,D_j) = \frac{\sum_{x \in (FIS(D_i,\alpha) \cap FIS(D_j,\alpha))} min\{supp(x,D_i), supp(x,D_j)\}}{\sum_{x \in (FIS(D_i,\alpha) \cup FIS(D_j,\alpha))} max\{supp(x,D_i), supp(x,D_j)\}} \quad (5)$$

Where α is the minimum support threshold, $FIS(D_i,\alpha)$ denotes the set of frequent itemsets reported from D_i and $supp(x,D_i)$ is the local support of *x* in D_i.

For each database pair (D_i,D_j), ($1 \le i < j \le n$), $sim_3(D_i,D_j)$ is computed and stored in a similarity table. As in [5], for each distinct ordered similarity value δ, a classification π^δ is generated and evaluated as follows.

7) Definition : The following goodness measure is used to assess each classification π^δ.

$$goodness(\pi^\delta) = \left| intra\_sim(\pi^\delta) + inter\_dist(\pi^\delta) - k \right| \quad (6)$$

Where $intra\_sim(\pi^\delta)$ is the intra-cluster similarity and $inter\_dist(\pi^\delta)$ is the inter-cluster distance and *k* is the number of clusters.

8) Definition : The intra-cluster similarity of π^δ is defined as follows.

$$intra\_sim(\pi^\delta) = \sum_{l=1}^{k} \sum_{D_i,D_j \in class^\delta_l}^{i \neq j} sim_3(D_i,D_j) \quad (7)$$

9) Definition : The inter-cluster distance of π^δ is defined as follows.

$$inter\_dist(\pi^\delta) = \sum_{class^\delta_l, class^\delta_h}^{l \neq h} \sum_{D_i \in class^\delta_l, D_j \in class^\delta_h}^{i \neq j} (1 - sim_3(D_i,D_j)) \quad (8)$$

The best classification is selected based on maximizing both the intra-cluster similarity and the inter-cluster distance. Hence, higher the value of *goodness*, better is the clustering. The proposed algorithm in [6] is optimal in terms of running time. It takes $O(m \times n^2)$ time to find the best classification, such that *n* is the database number and $m \in \left[1, \frac{n^2-n}{2}\right]$ is the number of all candidate classifications. The experiments carried out in [6] show that the proposed algorithm is effective and finds the optimal classification only after examining few similarity levels. However, the time complexity of the algorithm still needs to be optimized.

In 2013, Yaqiong LIU, Dingrong YUAN and Yuwei CUAN [7] have proposed a new algorithm for classifying databases based on the same similarity measure proposed by A.Adhikari, PR .Rao [6]. The proposed algorithm proceeds as follows. Initially each class contains one database object. These classes are created at level 1. Based on the classes of the level 1, classes of the level 2 are formed by merging the *i*-th class containing D_i with the *j*-th class containing D_j only if $sim_3(D_i, D_j) \ge \delta$. The algorithm continues further until no more class can be generated.

The proposed algorithm generates all the possible classes per level. It is simple but the clustering isn't effective when the database number increases. In fact, the time complexity of the algorithm is exponential $O(t \times 2^n \times n)$ such that *n* is the database number and *t* the distinct similarity values between the *n* databases.

IV. IMPROVING EXISTING MDC APPROACHES

The efficiency of a MDC approach depends mainly on the accuracy of its similarity measure and the time performance of the classification algorithm. Hence, improving the later parameters will lead to develop effective multi-database systems. In the following sections, we present some contributions to achieve our goals.

A. Enhancing the quality of the similarity measure

The similarity measure sim_3 proposed by A.Adhikari, PR .Rao in [6] has a certain limitation. In fact, the numerator of $sim_3(D_i,D_j)$ takes into account only the frequent itemset that are shared between two databases D_i and D_j and ignore patterns supported by only one database. When a pattern X is not frequent in D_i, that is, $X \notin FIS(D_i,\alpha)$, this doesn't mean that X is not present at all in D_i. It might be present with a certain support value less than the minimum support threshold α. Hence, X may be frequent when the two databases are integrated and a mono-database mining is applied. Therefore, we need to estimate the support of X in $\{D_i \cup D_j\}$ in case in which $supp(X,D_i)<\alpha$ or $supp(X,D_j)<\alpha$. To do so, we could use the synthesizing model proposed in [8] as follows.

10) Definition : The estimated support of X in the union $\{D_i \cup D_j\}$ is defined as follows [8].

$$supp_s(X, D_i \cup D_j) = \frac{supp(X, D_i) \times |D_i| + supp(X, D_j) \times |D_j|}{|D_i| + |D_j|} \quad (9)$$

Such that $supp(X,D_i)$ is the local support of X in D_i. We note by $|D_i|$ the number of transaction in D_i.

Inspired by the works proposed by A.Adhikari and PR .Rao in [6] and Ramkumar T, Srinivasan R in [8], we present the following similarity measure.

11) Definition : Let D_i and D_j be two transactional databases and α the minimum support value. $sim_4(D_i,D_j)$ is defined as follows.

$$sim_4(D_i, D_j) = \frac{\sum_{x \in (FIS(D_i,\alpha) \cup FIS(D_j,\alpha))} \Phi\{x, D_i \cup D_j, \alpha\}}{\sum_{x \in (FIS(D_i,\alpha) \cup FIS(D_j,\alpha))} max\{supp(x, D_i), supp(x, D_j)\}} \quad (10)$$

such that,

$$\Phi\{x, D_i \cup D_j, \alpha\} = \begin{cases} supp_s(X, D_i \cup D_j), if\ supp_s(X, D_i \cup D_j) \geq \alpha \\ 0, otherwise \end{cases} \quad (11)$$

The similarity function, sim_4, calculates the relevance between two databases based on the synthesized supports of the frequent itemsets obtained when the databases are mined together. The synthesized support of a frequent itemset represents the estimation of its real support obtained in case in which databases are integrated and a mono-database mining is applied. Thus, sim_4 will contribute to select the correct databases to analyze in order to enhance the quality of the discovered patterns.

Example : Let D_i and D_j be two databases , each of which has 10 transactions. Let the minimum support value be α=0.2. Then, the frequent itemsets obtained from each database are as follows.

$FIS(D_i,\alpha)=\{supp(A,D_i)=0.5, supp(B,D_i)=0.4, supp(C,D_i)=0.2\}$

$FIS(D_j,\alpha)=\{supp(C,D_j)=0.3\}$

We observe that A and B are not present in $FIS(D_j,\alpha)$ because their support values are less than α, with $supp(A,D_j)=0.1$ and $supp(B,D_j)=0.1$.

Let's calculate the similarity between D_i and D_j using the measure proposed in [6] : $sim_3(D_i,D_j)=\frac{0+0+0.2}{0.5+0.4+0.3}=0.167$
Now, by using our similarity measure we get the following result : $sim_4(D_i,D_j)=\frac{\frac{5+0}{20}+\frac{4+0}{20}+\frac{3+2}{20}}{0.5+0.4+0.3}=\frac{0.25+0.20+0.25}{0.5+0.4+0.3}=0.58$

The similarity measure proposed by A.Adhikari, PR .Rao in [6] has not taken into account the support value of A and B in D_j because ($supp(A,D_j)<\alpha$ and $supp(B,D_j)<\alpha$), consequently $sim_3(D_i,D_j)$ is too low. However, patterns A and B will be frequent when the two databases D_i and D_j are integrated and a frequent-itemset discovery is applied on the union $\{D_i \cup D_j\}$. In fact, the real support values of A and B in $\{D_i \cup D_j\}$ are calculated as follows : $supp(A, \{D_i \cup D_j\})=(5+1)/20=0.3$, $supp(B, \{D_i \cup D_j\})=(4+1)/20=0.25$. The values estimated in the numerator of our measure are close to the real values, with $supp_s(A, \{D_i \cup D_j\})=(5+0)/20=0.25$ and $supp_s(B, \{D_i \cup D_j\})=(4+0)/20=0.20$. Consequently, our similarity measure returns more accurate results.

sim_4, could be improved further more by using the correction factor h proposed in [9]. In fact, we could estimate the local support of X in D_i when $supp(X,D_i)<\alpha$ as follows.

12) Definition : The local support of X in D_i could be obtained as follows.

$$supp(X,D_i) = \begin{cases} h \times \alpha, if\ supp(X, D_i) < \alpha \\ supp(X, D_i), otherwise \end{cases} \quad (12)$$

Where $h \in [0,1]$ and α is the minimum support threshold value. Choosing an appropriate correction factor h is related to the data distributed in the database. More the subsets of X are frequent with a high support value, higher is the probability to see them appearing together and hence, h must be chosen close to 1 in this case. In the absence of such information, setting h=0.5 is a suitable choice. In the previous example, we could estimate the support of A and B in D_j by applying a correction factor h=0.5. Consequently, the following results are obtained : $supp(A,D_j)=supp(B,D_j)=0.5\times0.2=0.1$,which is exactly the real support value of A and B in D_j. Therefore, a new value of sim_4 is obtained too as follows :
$sim_4(D_i,D_j)=\frac{\frac{5+1}{20}+\frac{4+1}{20}+\frac{3+2}{20}}{0.5+0.4+0.3}=0.66$. As we can notice, our similarity measure indicates that D_i and D_j are relevant (with a similarity value above than 50%) since all the frequent itemsets from $\{FIS(D_i,\alpha) \cup FIS(D_j,\alpha)\}$ will be frequent in case

the two databases are integrated into one dataset and a mono-database technique is applied. Consequently, it is appropriate to put D_i and D_j into the same cluster and analyze them together to improve the quality of the discovered patterns.

B. *Improving the time performance*

According to the study presented in Section III, existing classification algorithms [4-7] generate hierarchical classifications that verify the following property.

13) Property: Let $class(D,sim,\delta_i)$ and $class(D,sim,\delta_j)$ be two candidate classifications of $D=\{D_1,D_2,\ldots,D_n\}$ generated at two consecutive similarity level δ_i and δ_j.Then, for each cluster g_x in $class(D,sim,\delta_i)$, there is certainly a cluster g_y in $class(D,sim,\delta_j)$, such that $g_x \subset g_y$

Despite of the latter property, the existing algorithms produce each classification independently, that is, instead to use the earlier clusters (e.i., generated in the previous classifications) to build the clusters of the next classification, they generate each classification starting from the initial state where each database forms one cluster $\{D_1\},\{D_2\},\ldots,\{D_n\}$. Consequently the existing algorithms are time consuming and do unnecessary work.

The above observations motivated us to design an improved MDC approach that overcomes the later problems as follows. We assume that each database has been mined using a fast algorithm for frequent itemset discovery [11]. Let $D=\{D_1,D_2,\ldots,D_n\}$ be the set of n transactional database objects to classify, then the problem of generating a classification $class(D,sim,\delta)$ at a similarity level δ, can be described in terms of determining the connected components of an undirected weighted graph $G=(D,E)$. We consider the database set $D=\{D_1,D_2,\ldots,D_n\}$ as the vertex set of G and E the edge set. The weight of an edge $(D_i,D_j) \in E$ is the similarity value between the corresponding databases $sim_4(D_i,D_j)$. At a certain similarity level δ, there is an edge connecting two vertices D_i and D_j if $sim_4(D_i,D_j) \geq \delta$.

Initially, the graph $G=(D,E)$ is empty with n disconnected vertices ,i.e, $E=\emptyset$. Then, similarities are calculated between each vertex pairs (D_i,D_j) using the similarity measure sim_4, for $i,j=1$ to n. After that, edges are added to E starting with the vertex pairs having the highest similarity as follows. Let $\delta_1,\delta_2,\ldots,\delta_m$ be the m distinct similarity values between the n databases sorted in the decreasing order (e.i., $\delta_1>\delta_2>\ldots>\delta_m$) and let $E_{\delta l}=\{(D_i,D_j)\in E,\ sim_4(D_i,D_j)=\delta_l,\ i,j=1 \text{ to } n,\ i\neq j\}$ be the list of edges with weight value equal to $\delta_l (l=1 \text{ to } m)$. For each distinct similarity value δ_l, a candidate classification $class(D,sim,\delta_l)$ is generated by adding the edge list $E_{\delta l}$ to the edge set E such that $E=E_{\delta 1}\cup E_{\delta 2}\cup\ldots\cup E_{\delta l-1}$.Then, it remains to identify the connected component of G in order to discover the database clusters in $class(D,sim,\delta_l)$.

To determine and maintain the connected component of G, we use a modified version of the disjoint-forest data structure [12]. We have presented the classification algorithm and data structures in details on another paper [13].

At a given similarity level δ_l, if each connected component of G, denoted $g^{\delta l}{}_k$ forms a clique (e.i., a subset of vertices, each pair of which is connected by an edge in E), then the corresponding classification $class(D,sim,\ \delta_l)=\{g^{\delta l}{}_1,\ g^{\delta l}{}_2,\ldots,\ g^{\delta l}{}_k\}$ is called a *complete* classification . For classification assessment, we use the *goodness* measure proposed in [6]. The complete classification with the maximum goodness value is selected as the best classification of the database set. Our classification approach is depicted in Fig. 1.

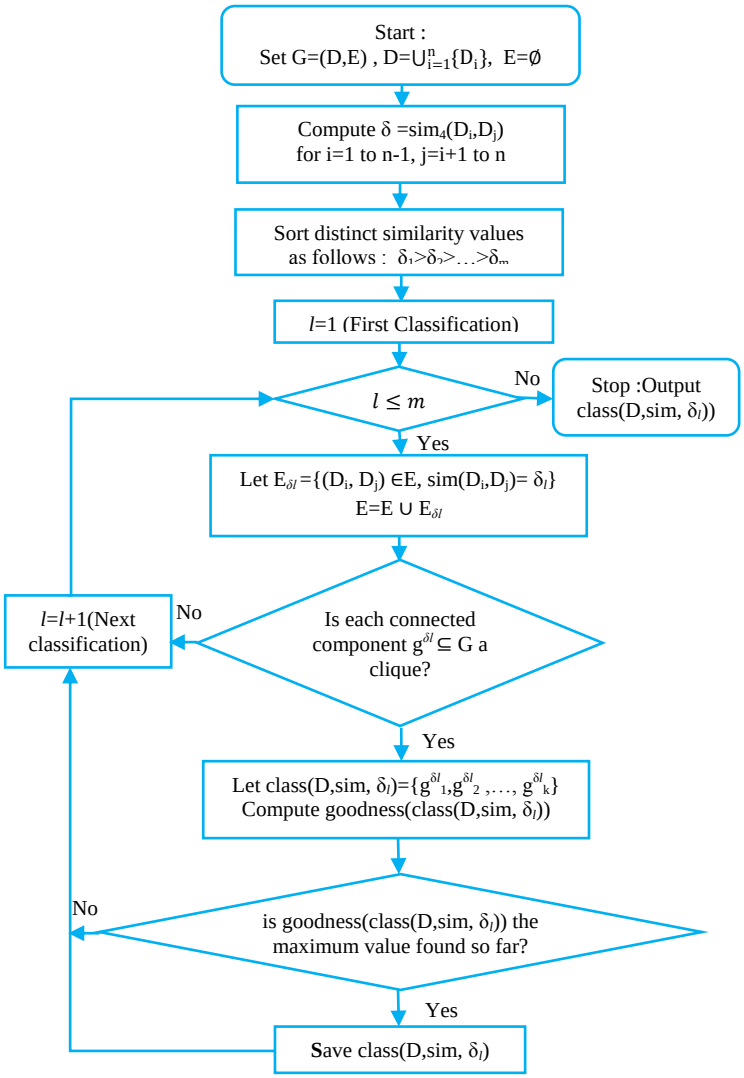

Fig. 1. Proposed approach for classifying the n multiple databases

C. *Performance Analysis*

To find the best classification of the n multiple databases, our algorithm implementing the proposed approach examines once all the m edge lists E_δ to generate and evaluate m candidate classifications. The average size of each edge list is n^2/m. Therefore, exploring all the m edge lists by our algorithm takes $O(n^2)$ time. Consequently, the proposed algorithm is optimal when comparing with *BestDatabasePartition*[6], which takes $O(m\times n^2)$ time to find the best classification of the n multiple databases such that $m \in \left[1,\frac{n^2-n}{2}\right]$. To assess the performance of our approach, we have conducted some experiments to compare the execution time obtained by *BestDatabasePartition*[6] and our graph-based classification algorithm. All the experiments have been implemented on a 2.70 GHz Pentium processor with 2 GB of

memory, using JAVA Edition 6. We carried out the experiments using two synthetic datasets T10I4D100K and T40I10D100K available in: http://fimi.ua.ac.be/data. For multi-database mining, each of the datasets is divided horizontally into 10 and 20 databases. The multiple databases obtained from T10I4D100K and T40I10D100K are referred to as $T_{1,1}$, $T_{1,2}$, …, $T_{1,n}$ and $T_{4,1}$, $T_{4,2}$, …, $T_{4,n}$ respectively, such that n=10, 20. By varying the value of the minimum support threshold, minsupp(α), we get different sets of frequent itemsets using FP-Growth algorithm [11]. Once the data-preparation done, we classify the multi-databases using our algorithm and *BestDatabasePartition*[6].

*1) First study :*In the first part of our experiments, we analyze the impact of minsupp(α) on the classification execution time. Therefore, we set the number of databases n=10 and we varied the value of α to get different sets of frequents itemsets. Then, we executed our algorithm and *BestDatabasePartition* on the same multiple database set. The execution times are presented in Fig. 2. From the experiment results, we observe that the execution time is relatively constant for the two algorithms, as the value of α increases.

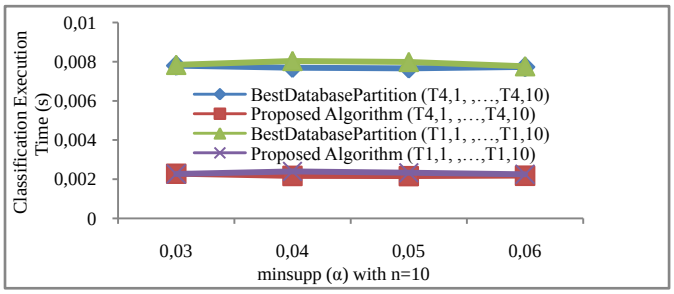

Fig. 2. Execution time versus minsupp

The reason is that n and the number of candidate classification m didn't change during the experiments (n=10, m=45). However, we notice that the classification time of our algorithm is shorter than that of *BestDatabasePartition* for α=0.03 to 0.06.

*2) Second study :*In the second part of our experiments, we studied how the number of databases n and the number of candidate classification m influence the time complexity of the two algorithms. Hence, we set the value of minuspp α=0.03 and we varied n from 3 to 20 databases. The experiment results obtained by the two algorithms are presented in Fig. 3

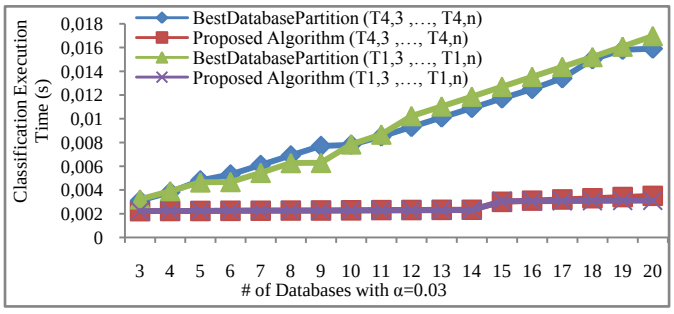

Fig. 3. Execution time versus the number of databases.

From the experiment results, we can see that the execution time tend go higher, as the value of n increases. However, we notice that the classification time increases faster for *BestDatabasePartition*. The reason is that m becomes larger as the value of n increases. Since the time complexity of *BestDatabasePartition* depends strongly on m, we note a rapid increase of *BestDatabasePartition*'s execution time for n=3 to 20. According to the above experiment results, our algorithm is the fastest. This is because, unlike the existing works[4-7], the time complexity of our algorithm depends only on the number of databases and takes $O(n^2)$ time to explorer all the m edge lists and find the best classification of the n multi-databases.

V. Conclusion and Future Work

Multi-database classification (MDC) approaches have been improved from based on a given application to independent-application. In this paper, we have discussed the existing MDC approaches by presenting their advantages and point out their limitations. Inspired by the existing works, we have proposed an improved approach to enhance the similarity measure and optimize the time performance of the existing classification algorithms. Experimental results have confirmed the efficiency of the proposed algorithm. Future work will be directed toward assessing our algorithm using real-world datasets and validate the tests on a real multi-databases system.

References

[1] Zhang, S., Zaki, M.J, "Mining multiple data sources: local pattern analysis," Data Mining and Knowledge Discovery. Springer, pp. 121–125, 2006.

[2] Zhang, S., Wu, X., Zhang, C, "Multi-database mining ," IEEE Comput. Intell. Bull. 2 (1), 5–13, 2003.

[3] Zhang S, Zhang C,Wu X, Knowledge discovery in multiple databases. Springer, New York, 2004.

[4] Wu X, Zhang C, Zhang S, "Database classification for multi-database mining ," Information Systems 30(1): 71–88, 2005.

[5] Li H, Hu X, Zhang Y, "An improved database classification algorithm for multi-database mining," in: Proceedings of the 3d International Workshop on Frontiers in Algorithmics, Springer, Berlin/Heidelberg, pp. 346–357 ,2009.

[6] A.Adhikari, PR .Rao, "Efficient clustering of databases induced by local patterns," Decision Support Systems 44(4):925–943, 2007.

[7] Yaqiong LIU, Dingrong YUAN, Yuwei CUAN, "Completely clustering for multi-databases mining," Journal of Computational Information Systems 9: 16, 2013.

[8] Ramkumar T, Srinivasan R, "Modified algorithm for synthesizing high-frequency rules from different data sources ," Knowledge and Information System 17(3) :313-334,2008.

[9] Thirunavukkarasu, R., Rengaramanujam, Srinivasan, "The Effect of Correction Factor in Synthesizing Global Rules in a Multi-databases Mining Scenario," Journal of computer science, 6(3): (2009).

[10] H. Liu, H. Lu, J. Yao, "Toward multi-database mining: identifying relevant databases," IEEE Trans. Knowledge Data Eng. 541–553,2001.

[11] Jiawi Han, Jina Pei, Yiwen Yin, Runying Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," Data Mining and Knowledge Discovery, pp. 53-87, 2000.

[12] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to algorithms. Cambridge, MA: MIT Press, 1990.

[13] Miloudi. S, Rahal S.A, Khiat. S, "Contribution to improve database classification algorithms for multi-Database mining," unpublished.