IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 4, 2020, accepted January 3, 2021, date of publication January 11, 2021, date of current version January 21, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050404

A Gradient-Based Clustering for Multi-Database Mining

SALIM MILOUDI, (Member, IEEE), YULIN WANG, (Senior Member, IEEE), AND WENJIA DING, (Member, IEEE)
School of Computer Science, Wuhan University, Wuhan 430072, China
Corresponding authors: Yulin Wang (yulinwang@whu.edu.cn) and Wenjia Ding (w.j.ding@whu.edu.cn)

ABSTRACT Multinational corporations have multiple databases distributed throughout their branches, which store millions of transactions per day. For business applications, identifying disjoint clusters of similar and relevant databases contributes to learning the common buying patterns among customers and also increases the profits by targeting potential clients in the future. This process is called clustering, which is an important unsupervised technique for big data mining. In this article, we present an effective approach to search for the optimal clustering of multiple transaction databases in a weighted undirected similarity graph. To assess the clustering quality, we use dual gradient descent to minimize a constrained quasi-convex loss function whose parameters will determine the edges needed to form the optimal database clusters in the graph. Therefore, finding the global minimum is guaranteed in a finite and short time compared with the existing non-convex objectives where all possible candidate clusterings are generated to find the ideal clustering. Moreover, our algorithm does not require specifying the number of clusters a priori and uses a disjoint-set forest data structure to maintain and keep track of the clusters as they are updated. Through a series of experiments on public data samples and precomputed similarity matrices, we show that our algorithm is more accurate and faster in practice than the existing clustering algorithms for multi-database mining.

INDEX TERMS Multi-database mining, graph clustering, dual gradient descent, quasi-convex optimization, similarity measure.

I. INTRODUCTION

The emergence of large multi-branch companies has led to developing new strategies for mining the transaction databases located at their different branches. To make decisions at a global level, the traditional process consists of integrating all the branch databases into a central repository called *data warehouse*, and then traditional mining algorithms [1]–[3] are applied on this huge accumulated dataset to discover the global patterns supported by all the branches. However, this approach may have some limitations related to (1) the cost of merging potentially heterogeneous databases, (2) the space and time complexity needed to mine and store the data warehouse and (3) the privacy issues preventing some branches from sharing their raw transactions. More importantly, some essential patterns could be disguised due to the integration of irrelevant data. To deal with the latter problems, the raw transaction data remain in place and the local *frequent itemsets* (FIs) mined at each branch are forwarded to a central site for clustering and aggregation.

Many clustering models have been proposed in the literature such as partitioning [4], hierarchical [5] and spectral-based models [6], which have applications in many fields including image segmentation [7], [8], social networks analysis and community discovery [9], [10], recommender systems [11]–[13] and so on [14], [15]. Clustering in the artificial neural networks (ANNs) literature is usually based on a competitive learning (CL) paradigm [16]–[18] where codebook weight vectors (prototypes) compete in order to elect the best matching unit (BMU), i.e., a neuron unit whose weight vector has the minimum distance to an input vector. Afterward, the selected prototype is updated to get closer to the input vector. Models and algorithms based on CL include vector quantization and self-organizing maps [19]–[21]. The CL algorithm continues to select and update the BMU until reaching a certain number of iterations. Despite their simplicity, there are some major limitations associated with CL algorithms, including sensitivity to initialization and difficulty of choosing an appropriate number of clusters beforehand.

The associate editor coordinating the review of this manuscript and approving it for publication was Keli Xiao.

In fact, due to inappropriate initialization, a BMU might win the competition too often. Consequently, the remaining weight vectors will never get updated, and hence engendering dead units. Moreover, to find the BMU of a given input vector, a distance between the input patterns and the prototypes must be defined, which is commonly the Euclidean distance. Since each transaction database is represented by its local frequent itemsets, metrics applied to vectors of real numbers (which are coordinates in a Euclidean space such as the Euclidean distance) cannot be applied in the case of computing distance between sets of frequent itemsets.

In this article, our study focuses on similarity-based clustering models for multi-database mining [22]–[25], due to their simplicity and stability [26]. In fact, similarity-based clustering represents a robust technique to partitioning graphs of n nodes into k sub-graphs of similar objects given a predefined similarity measure. However, the clustering evaluation measures used in the previous works [22]–[25] are non-convex functions, which makes finding the optimal clustering a difficult problem to solve without browsing all the local solutions (i.e., candidate clusterings generated at local optima). Moreover, sometimes the algorithms proposed in [22]–[25] fail to identify the ground truth clustering.

To address the limitations related to (1) sensitivity to cluster centers initialization and selection of the number of clusters k (i.e., requiring multiple restarts) and (2) optimization of non-convex clustering quality measures, we represent the problem of clustering multiple transaction databases as a quasi-convex optimization problem solvable without specifying the number of clusters beforehand. In contrast to competitive learning paradigm [16]–[18], we have adopted a gradient-based learning approach [27] with back-propagation [28] to minimize a clustering quasi-convex loss function $L(\theta)$ which guarantees convergence to the global minimum. We also discover the number of clusters (denoted by $f_\theta(\mathcal{D})$) in the input space by incorporating $f_\theta(\mathcal{D})$ into our objective.

Unlike logistic regression where hyperplanes or decision boundaries are learned to classify the training data given their respective class labels, and different than linear/polynomial regression where we learn the weighted model that best fits the training data given some actual continuous variable, our approach learns the optimal weights $\theta_{p,q}$ minimizing $L(\theta)$, such that $p = 0 \ldots n-2, q = p+1 \cdots n-1$, and n is the number of databases. From a graph-theoretic standpoint, each learned weight $\theta_{p,q}$ represents a discrete membership value that is associated with an edge in a similarity graph $G = (\mathcal{D}, E)$, where $\mathcal{D}$ is the vertex set consisting of the n database nodes and E is the edge set, which is initially empty.

In our proposed clustering model, we use a single-layer perceptron, where the input layer consists of the pairwise similarities $sim(\mathcal{D}_p, \mathcal{D}_q)$ to be fed into our model, and the output unit computes the intra-cluster similarity denoted by $W_\theta(\mathcal{D})$, which is then plugged into our loss function $L(\theta)$. In fact, the simplest and straightforward way to model the intra-cluster similarity is to use a weighted sum of the pairwise similarities where each learned weight $\theta_{p,q}$ decides whether the corresponding similarity value $sim(\mathcal{D}_p, \mathcal{D}_q)$ should take part in calculating $W_\theta(\mathcal{D})$. Precisely, if $\theta_{p,q} \geq 1$, an edge is added between $\mathcal{D}_p$ and $\mathcal{D}_q$ in G, $sim(\mathcal{D}_p, \mathcal{D}_q)$ is added up to $W_\theta(\mathcal{D})$, the number of clusters is decremented by one and the corresponding databases $\{\mathcal{D}_p, \mathcal{D}_q\}$ are put into the same cluster. Otherwise, if $\theta_{p,q} < 1$, we just keep updating the weights $\theta_{p,q}$ using gradient descent and back-propagation until we reach the global minimum. Our main contributions are:

- We propose a gradient-based approach to minimize a clustering quasi-convex loss function, which guarantees convergence to the global minimum.
- We reduce the running time of the recent multi-database clustering algorithms proposed in [22]–[25] by early stopping the clustering process at the global minimum of the objective function.
- We improve the accuracy of the existing similarity measure proposed in [22] by including the estimated supports of the infrequent itemsets.
- In our clustering method, the number of clusters denoted by $f_\theta(\mathcal{D})$ is not required beforehand. Instead, $f_\theta(\mathcal{D})$ becomes a parametric function in our loss function $L(\theta)$ that needs to be optimized as well.

Unlike the algorithms proposed in [22]–[25], by minimizing a quasi-convex loss function $L(\theta)$, our clustering algorithm does not need to generate and evaluate all the possible candidate clusterings in order to select the optimal one. Instead, we assess each clustering on the fly as it is generated and we terminate the generation process once we have reached the global minimum of the loss function. This is mainly possible because of the quasi-convexity of $L(\theta)$, i.e., while $L^{(i-1)} \geq L^{(i)}$, our clustering algorithm continues to add new edges and update the database clusters. Otherwise, the procedure terminates and returns the cluster labels at the iteration $i-1$.

The remainder of this article is organized as follows. In Section II, we present a motivating example and an overview of some existing clustering algorithms while pointing out their advantages and limitations. Section III describes some relevant concepts and then presents our approach to clustering multiple transaction databases using gradient descent and back-propagation. In Section IV, we perform several experiments to analyze and compare the proposed algorithm with the existing works in terms of both accuracy and running time. Finally, Section V concludes this article and highlights our future work.

II. MOTIVATION AND RELATED WORK

Prior to mining multiple databases for knowledge discovery, it is crucial to group these multiple databases into relevant and disjoint clusters sharing common patterns. Afterward, each cluster could be analyzed individually using techniques from [29]–[32] to identify new patterns such as the *high-vote patterns* [33] supported by all the branches and the *exceptional patterns* [34] supported by only few branches. The underlying

patterns are useful for making specific decisions on each group of branches operating under the same organization.

Example 1: Consider the following motivating example. Let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_6\}$ be a set of six transaction databases of a multi-branch company as defined in Table 1. Each branch database stores a set of customer transactions and each transaction enclosed in parentheses includes a list of items separated by commas. The sizes (i.e., number of transactions) corresponding to the six databases are 4, 5, 4, 3, 4 and 4, respectively. Let $\alpha = 0.5$ be the minimum support threshold set by the user. The local FIs mined in each local database are presented in Table 2, where each entry $\langle I_k, supp(I_k, \mathcal{D}_p)\rangle$ in $FIS(\mathcal{D}_p, \alpha)$ is a tuple of two elements, I_k, the name of the frequent itemset and $supp(I_k, \mathcal{D}_p) \in [0, 1]$, called the *support*, which is the frequency of I_k in $\mathcal{D}_p$.

TABLE 1. Transaction databases (TD).

TD	$Transactions$
$\mathcal{D}_1$	$(A, B, C, D), (B, C), (A, B, C), (A, C)$
$\mathcal{D}_2$	$(A, B), (A, C), (A, B, C), (B, C), (A, B, D)$
$\mathcal{D}_3$	$(B, C, D), (A, B, C), (B, C), (A, D)$
$\mathcal{D}_4$	$(F, G, H, I, J), (E, F, H), (F, H)$
$\mathcal{D}_5$	$(E, F, H, J), (F, H), (F, H, J), (E, J)$
$\mathcal{D}_6$	$(F, H, I, J), (E, H, J), (E, F, H), (E, I)$

TABLE 2. Frequent itemsets (FIs) mined from $\mathcal{D}_i$ in Table 1 under a minimum support threshold $\alpha = 0.5$ for $i = 1$ to $n = 6$.

TD	$FIS(\mathcal{D}_i, \alpha)$
$\mathcal{D}_1$	$\langle A, 0.75\rangle, \langle B, 0.75\rangle, \langle C, 1.0\rangle, \langle BC, 0.75\rangle, \langle ABC, 0.5\rangle, \langle AB, 0.5\rangle, \langle AC, 0.75\rangle$
$\mathcal{D}_2$	$\langle A, 0.8\rangle, \langle B, 0.8\rangle, \langle C, 0.6\rangle, \langle AB, 0.6\rangle$
$\mathcal{D}_3$	$\langle A, 0.5\rangle, \langle B, 0.75\rangle, \langle C, 0.75\rangle, \langle D, 0.5\rangle, \langle BC, 0.75\rangle$
$\mathcal{D}_4$	$\langle FH, 1.0\rangle, \langle F, 1.0\rangle, \langle H, 1.0\rangle$
$\mathcal{D}_5$	$\langle F, 0.75\rangle, \langle FH, 0.75\rangle, \langle H, 0.75\rangle, \langle FJ, 0.5\rangle, \langle FHJ, 0.5\rangle, \langle HJ, 0.5\rangle, \langle J, 0.75\rangle, \langle EJ, 0.5\rangle, \langle E, 0.5\rangle$
$\mathcal{D}_6$	$\langle H, 0.75\rangle, \langle EH, 0.5\rangle, \langle E, 0.75\rangle, \langle FH, 0.5\rangle, \langle F, 0.5\rangle, \langle HJ, 0.5\rangle, \langle J, 0.5\rangle, \langle I, 0.5\rangle$

Now, the global support of each itemset $I_k \in \cup_{p=1}^{6}\{FIS(\mathcal{D}_p, 0.5)\}$ in $\mathcal{D}$ is estimated using the synthesizing model proposed in [35] given as follows:

$$supp(I_k, \mathcal{D}) = (\sum_{p=1}^{n} |\mathcal{D}_p|)^{-1} \times \sum_{p=1}^{n} supp(I_k, \mathcal{D}_p) \times |\mathcal{D}_p| \quad (1)$$

such that $|\mathcal{D}_p|$ is the size of database $\mathcal{D}_p$ and n (6 in this example) is the number of all transaction databases in $\mathcal{D}$. For example, the global support of the itemset A in $\mathcal{D}$ is calculated as follows:

$$\begin{aligned} supp(A, \mathcal{D}) &= (\sum_{p=1}^{6} |\mathcal{D}_p|)^{-1} \times \sum_{p=1}^{6} supp(A, \mathcal{D}_p) \times |\mathcal{D}_p| \\ &= (4+5+4+3+4+4)^{-1} \\ &\quad \times (0.75 \times 4 + 0.8 \times 5 + 0.5 \times 4 + 0 \times 3 \\ &\quad + 0 \times 4 + 0 \times 4) = 0.375 \end{aligned}$$

The global supports of the rest of the itemsets are given in Table 3. As we can see in Table 3, all the global itemsets synthesized from the six databases are not valid or infrequent (i.e., $\forall\ I_k \in FIS(\mathcal{D}, 0.5), supp(I_k, \mathcal{D}) < \alpha$). As a result, no novel pattern has been found. This is mainly due to the irrelevant data involved during the synthesizing process. On the other hand, if we observe the transactions in the six databases, we may notice that they actually form two disjoint clusters, $C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ and $C_2 = \{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$, such that each cluster has similar transactions and common frequent itemsets.

Using the same model in (1) on the local frequent itemsets of each cluster separately ($p = 1 \cdots n = 3$ for C_1 and $p = 4 \cdots n = 6$ for C_2), we discover new valid frequent itemsets with a support $\geq \alpha$ in the clusters C_1 and C_2, that is, $FIS(C_1, 0.5) = \{\langle A, 0.692\rangle, \langle B, 0.769\rangle, \langle C, 0.769\rangle\}$ and $FIS(C_2, 0.5) = \{\langle H, 0.818\rangle, \langle F, 0.727\rangle, \langle F\ H, 0.727\rangle\}$. The frequent and infrequent patterns mined from each cluster are shown in Table 4. From the obtained results, we can conclude that more than 69% of the transactions in C_1 contain itemsets A, B and C, and more than 72% of the transactions in C_2 contain H, F and $F\ H$. Furthermore, other information could be derived from analyzing the association between frequent itemsets. For example, in the business sector, the itemset $\langle F\ H, 0.727\rangle \in FIS(C_2, 0.5)$ indicates that if a customer buys the item H, they are likely to also buy the item F with a confidence of $conf(H \rightarrow F, C_2) = \frac{supp(FH, C_2)}{supp(H, C_2)} = 88.87\%$ in at least one of the branches of C_2. Extracting such implications is also known as *association rule mining*, which has been studied extensively in the literature.

Based on the discovered patterns, the corporate headquarters will know what branches of the company share similar purchasing patterns, usually encompassing information such as frequency, quantity and timing. Also, the same business decisions and management strategies may be applied on the branches whose databases belong to the same cluster. Such decisions aim to reduce customer attrition, predict potential buying trends and convince clients to buy more products and services in the future. Therefore, analyzing the frequent itemsets coming from the same clusters contributes to identifying novel and useful patterns enhancing the quality of the decision making process.

Existing multi-database clustering algorithms [22], [23], [25], [36], [37] follow an agglomerative approach to produce hierarchical classifications at different similarity levels, such that each class in a given candidate classification is a subset of another class generated in the next classification. Despite

TABLE 3. Synthesized global itemsets from the union of the 6 databases in Table 1 under a minimum support threshold $\alpha = 0.5$.

I_k	$supp(I_k, \mathcal{D})$	I_k	$supp(I_k, \mathcal{D})$
$F\ J$	$0.083 < \alpha$	F	$0.33 < \alpha$
$F\ H$	$0.33 < \alpha$	$E\ H$	$0.083 < \alpha$
$F\ H\ J$	$0.083 < \alpha$	E	$0.208 < \alpha$
$H\ J$	$0.166 < \alpha$	D	$0.083 < \alpha$
J	$0.208 < \alpha$	C	$0.416 < \alpha$
$A\ C$	$0.125 < \alpha$	B	$0.416 < \alpha$
I	$0.083 < \alpha$	A	$0.375 < \alpha$
$A\ B$	$0.208 < \alpha$	$A\ B\ C$	$0.083 < \alpha$
H	$0.375 < \alpha$	$B\ C$	$0.25 < \alpha$
$E\ J$	$0.083 < \alpha$		

TABLE 4. Synthesized itemsets from the clusters C_1=$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ and C_2=$\{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$ under a minimum support threshold $\alpha = 0.5$.

$FIS(C_1, \alpha)$		$FIS(C_2, \alpha)$	
I_k	$supp(I_k, C_1)$	I_k	$supp(I_k, C_2)$
$\boldsymbol{A}$	$0.692 \geq \boldsymbol{\alpha}$	$H\ J$	$0.363 < \alpha$
$A\ C$	$0.230 < \alpha$	$E\ J$	$0.181 < \alpha$
$A\ B$	$0.384 < \alpha$	$E\ H$	$0.181 < \alpha$
$A\ B\ C$	$0.153 < \alpha$	J	$0.454 < \alpha$
$B\ C$	$0.461 < \alpha$	I	$0.181 < \alpha$
$\mathcal{D}$	$0.153 < \alpha$	$F\ J$	$0.181 < \alpha$
$\boldsymbol{C}$	$0.769 \geq \boldsymbol{\alpha}$	$\boldsymbol{H}$	$0.818 \geq \boldsymbol{\alpha}$
$\boldsymbol{B}$	$0.769 \geq \boldsymbol{\alpha}$	$F\ H\ J$	$0.181 < \alpha$
		$\boldsymbol{F\ H}$	$0.727 \geq \boldsymbol{\alpha}$
		$\boldsymbol{F}$	$0.727 \geq \boldsymbol{\alpha}$
		E	$0.454 < \alpha$

the latter property, these algorithms generate each candidate classification independently, that is, instead of using the earlier clusters generated previously to build new ones, they generate each classification starting from the initial state where each database forms its own cluster. Consequently, the existing algorithms in [22], [23], [25], [36], [37] execute an unnecessary work by reconstructing already existing clusters.

The above observations motivated the authors in [24] to propose a graph-based algorithm that keeps track of the earlier clusters formed in the previous levels and use them to form the new subsequent clusters. Compared with the previous works in [22], [23], [25], [36], [37], the clustering algorithm proposed in [24] is much faster in practice. However, it is still based on the clustering evaluation measure proposed in [22], which is a non-convex function whose global minimum is NP-hard to find. To deal with this main limitation, we transform the problem of clustering multiple transaction databases into a quasi-convex optimization problem where stochastic gradient descent can certainly converge to the global minimum. This allows us to stop the clustering process earlier without exploring the remaining candidate classifications. Hence, we can generate an optimal clustering in a reduced execution time.

Although k-means [4], [38] is a popular intuitive clustering approach that is easy to implement, it has the disadvantage of being sensitive to both centroids and number of clusters initialization. In fact, due to the non-convexity of its objective function, k-means cannot handle non-convex sets and only works well on data described by spatially separated hyperspheres. Despite the fact that we can select the number of clusters through interpretation of the Silhouette plot [39] (i.e., a technique that quantifies the quality of the clustering), this method requires running k-means on the same data for each number of clusters $k \in \{2, 3, 4 \ldots n-1\}$ in order to determine the optimal clustering that maximizes the Silhouette coefficient. Consequently, this might influence the time performance of the algorithm on large high dimensional datasets.

In our clustering approach, the optimization problem is quasi-convex and the global optimum can therefore be found independently of the initial settings. Moreover, our method does not require the number of clusters to be specified beforehand. Instead, the number of clusters becomes a parametric function in our loss that needs to be optimized as well. Furthermore, as it was pointed out in [22], traditional clustering algorithm such us BIRCH [40], k-means [4], [38] and Hierarchical clustering [5] are based on metric attributes whose values are represented by explicit coordinates in a Euclidean space. In such configuration, two similar multi-dimensional data points must have a small Euclidean distance value. Therefore, a traditional algorithm may not function properly, since we are interested in clustering local frequent itemsets from multiple databases.

The k-means algorithm converges to a local minimum solution [41] (due to sensitivity to the initial seeds and non-convexity of the objective function), which is not necessarily the optimal solution. In fact, the utilization of a random start procedure might produce slightly different results for every run. Consequently, many clustering algorithms attempt to obtain a good approximate grouping, which tends to be a reasonable solution in practice. On the other hand, Hierarchical clustering avoids the issue of presetting the number of clusters k beforehand by visualizing the relationships between objects at different nested hierarchical levels called *dendrogram*. This allows the domain expert to cut the tree diagram at any desired number of clusters. However, assessing the stability of a chosen cut-point allowing us to obtain the desired number of clusters remains difficult and dependent on some non-convex evaluation metrics. Moreover, the traditional algorithm for hierarchical clustering takes $O(n^3)$ time, which is not very efficient in practice.

Additionally, Hierarchical clustering faces some limitations when dealing with high-dimensional feature-sets whose components are not coordinates in a d-dimensional Euclidean space. This is due to the fact that we cannot calculate the cluster centroid of frequent itemsets or patterns which are not real numbers, and therefore cannot be located in a Euclidean space using the Euclidean distance. Also, because of the size and sparsity of the one-hot encoded feature vectors, patterns

in databases should be kept as sets or hash-tables, due to the huge size of the vocabulary containing the unique frequent itemsets coming from the n databases.

To overcome the limitations of single clustering algorithms such as sensitivity to cluster centroids initialization, sampling variability and prior selection of the number of clusters, a robust clustering algorithm, namely *Ensemble Clustering* (EC) [42]–[45], might be used to first generate multiple base clusterings which are then aggregated and summarized into a final optimal clustering result. However, running a single or different clustering algorithms on a resampled feature space of the dataset for few iterations and for each preselected number of clusters k might increase the running time overhead to find the best clustering. Moreover, the sensitivity of EC algorithms have not been systemically assessed. Additionally, the authors in [46] have performed some simulations and found that common implementations of EC perform poorly in determining the ground-truth number of clusters for data with known structure. They also suggested to apply and interpret EC with caution.

The authors in [47] have proposed an algorithm to accelerate mining global frequent itemsets (FIs) in large databases by adopting a divide-and-conquer approach. This approach is also referred to as *mono-database mining*. Similar works have been proposed in [48], [49]. First, the whole transaction database is partitioned into k non-overlapping partitions of similar transactions small enough to fit in main memory. Then, each partition is read one at a time and mined for local frequent itemset discovery. Afterward, all the local FIs from the k partitions are aggregated into global patterns using a merging function such as the aggregation model proposed in [35]. In the case of mono-database mining, one can directly access the raw transactions for pattern discovery. However, in the case of multi-database mining (MDM), the local data is left in place due to data privacy, and the patterns that has already been mined locally at each branch are forwarded to a central site for clustering and analysis. This not only allows us to preserve the safety of certain data which need to be kept confidential for competitive reasons but also reduces the cost of moving huge raw data over a communication network. In such scenario, the task of mining the local patterns is managed by each individual branch locally.

Unlike clustering the transactions in one database, our study focuses on clustering the local frequent itemsets coming from n multiple branch data sources and without specifying the number of clusters beforehand. This aims to discover the optimal database clusters sharing the same relevant patterns which are then used to take regional decisions regarding the branches exhibiting similar characteristics. In fact, the local FIs in each cluster are analyzed individually to discover new patterns such as *high-vote patterns* [33] and *exceptional patterns* [34], [50], [51] that are useful for making special decisions regarding some branches. In the partitioning approach [47]–[49], we can only discover the global patterns that are supported by the whole organization.

III. MATERIALS AND METHODS

In this section, we describe our gradient-based clustering approach. Some relevant concepts and definitions are also presented to understand the main concepts used in this article.

A. PROBLEM STATEMENT

Let $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_2, \ldots, \mathcal{D}_{n-1}\}$ be a set of n transaction databases, each located at one of the n different branches of a large company. Segmenting $\mathcal{D}$ consists of finding the optimal grouping of disjoint homogeneous database clusters that has the largest intra-cluster similarity and the largest inter-cluster distance possible. Let $G = (\mathcal{D}, E)$ be a similarity graph, such that $\mathcal{D}$ represents the vertex set and E represents the edge set, which is initially empty, i.e., $E \leftarrow \varnothing$. A similarity-based clustering algorithm proceeds in two steps:

1) Given a similarity measure *sim*, the algorithm computes $\binom{n}{2}$ similarity values between the n databases.
2) Edges $(\mathcal{D}_p, \mathcal{D}_q)$ are sorted in the non-increasing order of their corresponding similarities $sim(\mathcal{D}_p, \mathcal{D}_q)$ and added gradually to the graph G, such that at a similarity level δ, there is an edge between two nodes $\mathcal{D}_p$ and $\mathcal{D}_q$ if and only if $sim(\mathcal{D}_p, \mathcal{D}_q) \geq \delta$

Different candidate clusterings could be obtained by varying the similarity level δ. Therefore, a goodness measure is needed to evaluate each candidate clustering to see whether it satisfies a predefined optimality criterion. Depending on its convexity for a minimization problem or on its concavity for a maximization problem, the main purpose is to define an objective parametric function $L(\theta)$ in such a way that finding the optimum of $L(\theta)$ results in a clustering which has the largest within-cluster similarity and the largest between-cluster distance possible, while maintaining the number of clusters smaller than n and larger than one.

B. BACKGROUND AND RELEVANT CONCEPTS

In this subsection, we define some relevant concepts related to mining frequent itemsets in transaction databases and propose an encoding to efficiently represent the feature-set of each database. We also define the scalar function used to compute the similarity between two databases. Other definitions related to the intra-cluster similarity and the inter-cluster distance are also presented hereafter.

1) TRANSACTION DATABASE

A transaction database, denoted by $\mathcal{D}_p$, is usually collected by online transaction processing (OLTP) systems, which capture, store, and process thousands of transactions per second. In fact, transaction/OLTP data sources could be in different formats: relational (Oracle, SQL Server, MySQL, etc.), non-relational databases (MongoDB, Cassandra, etc.) or flat files (e.g., delimited text files (.txt), comma separated values text files (.csv), etc.). Generally, a transaction in $\mathcal{D}_p$ includes a unique ID number and a list of items. Depending on the application, these items could be products purchased in one trip (a shopping basket), words in a document, side effects

of drugs on certain patients, base pairs in genes, clicks on a particular web page, machine components broken down on a certain date, etc. For data mining, analytics and decision-making purposes, the data from OLTP operational sources is imported into online analytical processing (OLAP) systems (referred to as data warehouse) through a process named extract, transform, load (ETL). Depending on the decisional level, we can find two types of data warehouses:

- Enterprise Data Warehouse: it consists of a central repository which may be mined to discover global patterns useful for making global decisions on the whole enterprise.
- Data Mart: which represents a cluster of some relevant operational sources sharing similar characteristics such as customers exhibiting the same purchasing behavior. Mining a data mart is useful in identifying regional patterns necessary for making decisions specific to a certain area or a business unit.

2) FREQUENT ITEMSET MINING

Let $\mathcal{D}_p$ be a transaction database. A candidate itemset I_k of size k from $\mathcal{D}_p$ is a set of k items, defined as $I_k = \{x_1, x_2, \cdots, x_k\}$. The support of I_k in $\mathcal{D}_p$, defined as $supp(I_k, \mathcal{D}_p) = |\{T \in \mathcal{D}_p \mid I_k \subseteq T\}| \div |\mathcal{D}_p|$, is the proportion of transactions T in $\mathcal{D}_p$ that include I_k. An itemset I_k is *frequent*, if its support value is greater than or equal to some user-specified minimum support $\alpha \in [0, 1]$, i.e., $supp(I_k, \mathcal{D}_p) \geq \alpha$. Let n be the number of all unique items in $\mathcal{D}_p$. The number of all candidate itemsets is equal to $2^n - 1$.

Many algorithms [1]–[3], [52]–[54] have been proposed in the literature to discover frequent patterns in transaction databases. Generally, these algorithms adopt certain strategies to reduce the number of all candidate itemsets (e.g., using the downward closure), while making the least number of passes over the whole dataset stored in the disk (two scans at most over the data). For this purpose, efficient tree data structures have been used (e.g., FP-tree structure) to compact and encode the database by mapping each transaction onto a path in the tree residing in the main memory. Due to its good performance in terms of CPU and I/O overheads, we use FP-Growth [3] in all our experiments to mine frequent itemsets in the n local databases $\mathcal{D}_p, p = 0 \ldots n-1$.

3) MULTI-DATABASE MINING FOR PATTERN DISCOVERY

Given n transaction databases $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_{n-1}\}$ located at the n branches of a large company, the multi-database mining (MDBM) on $\mathcal{D}$ is carried out following four steps:

1) Forwarding the local frequent itemsets $FIS(\mathcal{D}_p, \alpha_p)$ mined in each local database $\mathcal{D}_p$ at a given minimum threshold α_p to a centralized site for clustering.
2) Clustering $\cup_{p=0}^{n-1}\{FIS(\mathcal{D}_p, \alpha_p)\}$ into non-overlapping clusters $\{C_1, C_2, \cdots, C_k\}$ such that:
   - $\cup_{i=1}^{k} C_i = \mathcal{D}$ and $\cap_{i=1}^{k} C_i = \emptyset$
   - $\forall\, \mathcal{D}_p, \mathcal{D}_q \in C_i, sim(\mathcal{D}_p, \mathcal{D}_q) \geq \delta$
   - $\forall\, \mathcal{D}_p \in C_i, \forall\, \mathcal{D}_q \in C_j, sim(\mathcal{D}_p, \mathcal{D}_q) < \delta$
   - A clustering quality measure $goodness(\mathcal{D})$ reaches its optimum at a similarity threshold δ.
3) Identifying new patterns by analyzing the local frequent itemsets [29], [55], [56] in each individual cluster C_i such as the *high-vote patterns* [33] highly supported by all the databases within one cluster and the *exceptional patterns* [34], [50], [51] supported by only few databases.
4) Synthesizing the global patterns from each cluster using one of the various weighting models proposed in the literature [30], [35], [57], [58].

In [35], the authors have proposed a method to synthesize high frequency patterns from different data sources having different sizes based on the number of transactions in each local database. Using their method, the obtained results have closely matched the global patterns discovered when performing a mono-database mining on the union of all the data sources. Therefore, the synthesized global support of an itemset I_k in a given cluster $C = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_m\}$ is defined as follows:

$$supp(I_k, \mathcal{C}) = \Big(\sum_{p=1}^{m} |\mathcal{D}_p|\Big)^{-1} \times \sum_{p=1}^{m} supp(I_k, \mathcal{D}_p) \times |\mathcal{D}_p| \quad (2)$$

such that $1 \leq m \leq n$ and $supp(I_k, \mathcal{D}_p)$ is the local support of I_k in the p-th database. We note that $|\mathcal{D}_p|$ and $|\mathcal{D}_p| / \sum_{p=1}^{m} |\mathcal{D}_p|$ are the weight (number of transactions) and the normalized weight of the transaction database localed at branch p, respectively. The notion of assigning a weight to each local database has been largely studied in the literature [35], [57], [59]. This allows the different branches taking part in the decision making process to have a weight, also called a vote, proportionally related to the number of transactions accumulated in their local databases.

4) SIMILARITY MEASURE

Each database $\mathcal{D}_p$ is encoded as a hash-table $FIS(\mathcal{D}_p, \alpha) = \{\cup_{k=1}^{m} \langle I_k, supp(I_k, \mathcal{D}_p)\rangle\}$ for all $p = 0 \ldots n-1$, where n is the number of transaction databases, m is the number of frequent itemsets or entries in $FIS(\mathcal{D}_p, \alpha)$, I_k is the key or the name of the k-th frequent itemset and $supp(I_k, \mathcal{D}_p) \in [0, 1]$ is the associated support value, which is the proportion of rows in $\mathcal{D}_p$ containing I_k and $\alpha \in [0, 1]$ is the minimum support threshold, such that $supp(I_k, \mathcal{D}_p) \geq \alpha$. When computing the similarity between two databases, it is practical to use hash-tables to efficiently lookup the keys (i.e., the itemsets) shared in common. Although key collisions may occur, using a hash function that is simple to compute and which distributes the keys uniformly on the buckets of the hash-table will allow us, in the average case, to check for the existence of a given key in a constant time $O(1)$. In our experiments, Python hash built-in function is used for key duplicate search.

In the context of multi-database mining, the goal is to put the databases that share a large number of frequent itemsets

having close support values into the same cluster. Inspired by the similarity function proposed in [22], our method consists of using the correction factor $cf \in [0,1)$ proposed in [60] to estimate the support values of the infrequent itemsets (i.e., itemsets whose supports are under the minimum threshold α), leading to a more accurate similarity measure. Hence, the similarity between two databases $\mathcal{D}_p$ and $\mathcal{D}_p$ for $p = 0 \ldots n-2, q = p+1 \ldots n-1$ is defined as follows:

$$sim(\mathcal{D}_p, \mathcal{D}_q) = \frac{\sum_{I_k \in \{FIS(\mathcal{D}_p,\alpha) \cup FIS(\mathcal{D}_q,\alpha)\}} \boldsymbol{min}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q))}{\sum_{I_k \in \{FIS(\mathcal{D}_p,\alpha) \cup FIS(\mathcal{D}_q,\alpha)\}} \boldsymbol{max}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q))} \quad (3)$$

where

$$\Psi(I_k, \mathcal{D}_p) = \begin{cases} supp(I_k, \mathcal{D}_p) & if\ I_k \in FIS(\mathcal{D}_p, \alpha) \\ (i.e.,\ supp(I_k,\ \mathcal{D}_p) \geq \alpha); & \\ cf \times \alpha & otherwise\ cf \in [0, 1) \end{cases}$$

When $cf = 0$, we get the same similarity measure proposed in [22].

Theorem 1: The similarity function $sim(\mathcal{D}_p, \mathcal{D}_q)$ has the following properties:

1) $0 \leq sim(\mathcal{D}_p, \mathcal{D}_q) \leq 1$ *(non-negativity)*
2) $sim(\mathcal{D}_p, \mathcal{D}_q) = sim(\mathcal{D}_q, \mathcal{D}_p)$ *(symmetry)*
3) $sim(\mathcal{D}_p, \mathcal{D}_p) = 1$ *(identity)*
4) $0 \leq (1 - sim(\mathcal{D}_p, \mathcal{D}_q)) = dist(\mathcal{D}_p, \mathcal{D}_q) \leq 1$

Proof: For each $I_k \in \{FIS(\mathcal{D}_p, \alpha) \cup FIS(\mathcal{D}_q, \alpha)\}$, $\boldsymbol{min}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q)) \leq \boldsymbol{max}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q))$. Therefore:

$$0 \leq \sum_{I_k} \boldsymbol{min}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q)) \leq \sum_{I_k} \boldsymbol{max}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q)) \leq \sum_{I_k} 1$$

By dividing each side of the inequality by $\sum_{I_k} \boldsymbol{max}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_q))$, we get property (1). Since ***min*** and ***max*** are commutative operations (i.e., $\boldsymbol{min}(x, y) = \boldsymbol{min}(y, x)$ and $\boldsymbol{max}(x, y) = \boldsymbol{max}(y, x)$), property (2) is satisfied. Property (3) is valid because $\boldsymbol{min}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_p)) = \boldsymbol{max}(\Psi(I_k, \mathcal{D}_p), \Psi(I_k, \mathcal{D}_p))$ for the same database $\mathcal{D}_p$. For property (4), we have $0 \leq sim(\mathcal{D}_p, \mathcal{D}_q) \leq 1 \xrightarrow{yields} -1 \leq -sim(\mathcal{D}_p, \mathcal{D}_q) \leq 0 \xrightarrow{yields} 0 \leq 1 - sim(\mathcal{D}_p, \mathcal{D}_q) \leq 1 \xrightarrow{yields} 0 \leq dist(\mathcal{D}_p, \mathcal{D}_q) \leq 1$ □

Each transaction database object $\mathcal{D}_p$ is represented as a bag of words, where each word is a frequent itemset I_k associated with a nonnegative weight, i.e., $\Psi(I_k, \mathcal{D}_p) = supp(I_k, \mathcal{D}_p) \in [\alpha, 1]$ if $I_k \in FIS(\mathcal{D}_p, \alpha)$ and $\Psi(I_k, \mathcal{D}_p) = (cf \times \alpha) \in [0, \alpha)$, otherwise. Note that if we replace all the weights $\Psi(I_k, \mathcal{D}_p)$ with binary values, either one if $I_k \in FIS(\mathcal{D}_p, \alpha)$ or zero otherwise, $sim(\mathcal{D}_p, \mathcal{D}_q)$ just simplifies to the Jaccard similarity. Unlike the similarity measure proposed in [22], when an itemset I_k is infrequent in $\mathcal{D}_p$, i.e., $\langle I_k, supp(I_k, \mathcal{D}_p)\rangle \notin FIS(\mathcal{D}_p, \alpha)$, we do not assume that it has zero support. In fact, an infrequent itemset may exist in the database with some frequency below the minimum threshold α.

To demonstrate the importance of estimating the support of infrequent itemsets in our similarity measure, we have the following example. Let $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ be three databases with their number of transactions 10,000, 30,000 and 10,000, respectively. The support values of their corresponding itemsets are presented in Table 5. Let the minimum support value be $\alpha = 0.20$. Itemset AC is not reported from $\mathcal{D}_3$ since its support is less than α, with $supp(A\ C, \mathcal{D}_3) = 0.15$. Also, the itemsets $A\ D$ and $C\ E$ are not reported from $\mathcal{D}_1$, because $supp(A\ D, \mathcal{D}_1) = 0.08 < \alpha$ and $supp(C\ E, \mathcal{D}_1) = 0.07 < \alpha$. Now, if we calculate the similarity between $\mathcal{D}_1$ and $\mathcal{D}_3$ using measure (3) with a correction factor $cf = 0$, we get $sim(\mathcal{D}_1, \mathcal{D}_3) = \frac{0.0+0.0+0.0}{0.30+0.40+0.50} = 0$. Since *sim* proposed in [22] takes into account only the frequent itemsets shared between two databases, and in this case $A\ C$, $A\ D$ and $C\ E$ are not common to both databases $\mathcal{D}_1$ and $\mathcal{D}_3$, then the numerator of *sim* will be nil. However, $A\ C$, $A\ D$ and $C\ E$ do really exist in the previous databases and we should have estimated their support values to get more accurate similarities.

By using a correction factor $cf = 0.5$, the estimated support values are obtained as follows: $supp(A\ C, \mathcal{D}_3) = supp(A\ D, \mathcal{D}_1) = supp(C\ E, \mathcal{D}_1) = cf \times \alpha = 0.1$, which are approximately close to the real values. Now, if we calculate the similarity between $\mathcal{D}_1$ and $\mathcal{D}_3$ while including the estimated support values, we get $sim(\mathcal{D}_1, \mathcal{D}_3) = \frac{0.1+0.1+0.1}{0.30+0.40+0.50} = 0.25$. The similarities between the remaining databases are presented in Table 6 with a correction factor $cf = 0$ (i.e., as proposed in [22]) and $cf = 0.5$ under our measure (3). After clustering the three databases using the algorithm proposed in [24] with a correction factor $cf = 0$ and $cf = 0.5$, we got the results presented in Table 7. We notice that $\mathcal{D}_1$ does not belong to the cluster $\{\mathcal{D}_2, \mathcal{D}_3\}$ when $cf = 0$ because $sim(\mathcal{D}_1, \mathcal{D}_3) = 0$. After using $cf = 0.5$, $\mathcal{D}_1$ is now added to $\{\mathcal{D}_2, \mathcal{D}_3\}$ and *goodness* [22] of $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ is larger than that of $\{\mathcal{D}_1\}$, $\{\mathcal{D}_2, \mathcal{D}_3\}$. Consequently, the proposed similarity measure in (3) returns more accurate results when using a correction factor $cf > 0$.

Choosing an appropriate value for cf is related to the itemsets distribution in the database. In other words, the more subsets of an itemset I_k are frequent (i.e., $\forall x_i \subseteq I_k, supp(x_i, \mathcal{D}_p) \geq \alpha$) with a large support value ≈ 1, the higher the probability to find these frequent subsets appearing together in the same rows of $\mathcal{D}_p$. In this case, cf must be chosen close to 1. When there is no prior information on the data distribution, setting $cf = 0.5$ is a suitable choice in practice. To estimate the support values of infrequent patterns, the authors in [61], [62] have used probabilistic models based on *maximum entropy* and *Markov random field* to predict the number of rows in the data containing a given itemset. However, these models need to compute all the combinations of sub-pattern frequencies which can be time consuming especially when the sizes and number of the infrequent patterns are large. Hereafter, we present three methods to estimate and compute the support of an infrequent itemset I_k given some of its frequent sub-sets.

Method 1. Since the support of an infrequent k-itemset I_k is below the minimum support threshold $< \alpha$, we can estimate the support of I by subtracting a certain value from α. This value is mainly dependent on the distribution of the frequent subsets $x_i \subseteq I_k$ of size $< k$. That is, the more the subsets of an itemset I_k are frequent (i.e., $\forall x_i \subseteq I_k, supp(x_i, \mathcal{D}_p) \geq \alpha$) with a large support value ≈ 1, the higher the probability to find these frequent subsets appearing together in the same rows of $\mathcal{D}_p$, and in this case a small value should be taken out from α. Conversely, the less the frequent subsets of I_k, the larger the value that should be taken out from α. Therefore, we can use the following heuristic to estimate the support of an infrequent k-itemset using the supports of its frequent subsets:

$$supp(I_k, \mathcal{D}_p) = \alpha \times \left(1 - \frac{\binom{k}{l} - \sum_{x_i \subseteq I_k, |x_i|=l} supp(x_i, \mathcal{D}_p)}{\binom{k}{l}}\right) \tag{4}$$

TABLE 5. Transaction databases with their corresponding itemsets under the minimum support threshold $\alpha = 0.20$.

Database name	Number of rows	Supports of the itemsets		
		$A\ C$	$A\ D$	$C\ E$
$\mathcal{D}_1$	10,000	0.30	**0.08**	**0.07**
$\mathcal{D}_2$	30,000	0.40	0.35	0.40
$\mathcal{D}_3$	10,000	**0.15**	0.40	0.50

TABLE 6. Similarity matrix between the three databases from Table 5 with $cf = 0$ as proposed in [22] against $cf = 0.5$ under our similarity measure *sim* (3).

$sim(\mathcal{D}_p, \mathcal{D}_q)$	$\mathcal{D}_1$		$\mathcal{D}_2$		$\mathcal{D}_3$	
	$cf = 0$	$cf = 0.5$	$cf = 0$	$cf = 0.5$	$cf = 0$	$cf = 0.5$
$\mathcal{D}_1$	1	1	0.26	0.43	0	0.25
$\mathcal{D}_2$	0.26	0.43	1	1	0.57	0.65
$\mathcal{D}_3$	0	0.25	0.57	0.65	1	1

TABLE 7. Clustering of the three databases from Table 5 with $cf = 0$ as proposed in [22] against $cf = 0.5$ under our similarity measure *sim* (3).

Output	Clustering under $cf = 0$	Clustering under $cf = 0.5$
clusters	$\{\mathcal{D}_1\}, \{\mathcal{D}_2, \mathcal{D}_3\}$	$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$
intra-cluster similarity	0.57	1.33
inter-cluster distance	1.74	0.0
goodness [22]	0.31	0.33

such that, at the l-th level ($k - 1 \geq l \geq 1$), there are $\binom{k}{l}$ sub-itemsets of size l. In (4), l takes the value of the level where at least one sub-itemset is found frequent. For example, let us consider the infrequent itemset $I_k = ABC$ of size $k = 3$, such that $supp(ABC, \mathcal{D}_p) < \alpha$ (e.g., $\alpha = 0.2$). we notice that ABC has $\binom{3}{2} = 3$ sub-itemsets of size $l = 2$, namely, $\{AB, AC, BC\}$, and also has $\binom{3}{1} = 3$ sub-itemsets of size $l = 1$, $\{A, B, C\}$. If the first two frequent subsets of size $< k$ are AB and AC with $supp(AB, \mathcal{D}_p) = 0.21$ and $supp(AC, \mathcal{D}_p) = 0.3$, then $l = 2$ and $supp(ABC, \mathcal{D}_p) = 0.2 \times (1 - \frac{3-(0.21+0.3+0)}{3}) = 0.034$. If we suppose that AB and AC have higher supports with $supp(AB, \mathcal{D}_p) = 0.4$ and $supp(AC, \mathcal{D}_p) = 0.5$, then the estimated support should be even closer to α, that is, $supp(ABC, \mathcal{D}_p) = 0.2 \times (1 - \frac{3-(0.4+0.5+0)}{3}) = 0.06$. If at $l = 2$, no frequent 2-subset is found, then we check the subsets at the lower level, i.e., at $l = 1$. If all the $2^k - 2$ subsets are infrequent, then $supp(I_k, \mathcal{D}_p) = 0$.

Method 2. We can also approach this estimation problem by examining the support values of the frequent subsets. Consider the same infrequent 3-itemset ABC. Let us suppose $\alpha = 0.2$ and the subsets AB and AC are frequent, such that $supp(AB, \mathcal{D}_p) = 0.7$ and $supp(AC, \mathcal{D}_p) = 0.4$, respectively. Clearly, since ABC is infrequent (i.e., $ABC \notin FIS(\mathcal{D}_p, \alpha)$), both AB and AC appear together with a percentage of less than 20% of the transactions in $\mathcal{D}_p$ (i.e., $supp(ABC, \mathcal{D}_p) < 20\%$). Also, the percentage of transactions where AB does not occur is $1 - 0.7 = 30\%$. Since $supp(AC, \mathcal{D}_p) = 0.4 > 30\%$, then at least $0.4 - 0.3 = 10\%$ of the transactions must be shared between AB and AC, which means the possibility that $supp(ABC, \mathcal{D}_p) = 0$ is excluded. Therefore, in order to estimate the support of ABC, we can calculate $supp(ABC, \mathcal{D}_p) = 10\% + (\alpha - 10\%) \times \epsilon$, where $\epsilon \in [0, 1)$ is selected by the domain expert. If $supp(AC, \mathcal{D}_p) < 30\%$, then $supp(ABC, \mathcal{D}_p)$ could be equal to zero, and therefore setting $supp(ABC, \mathcal{D}_p) = (\alpha \times 0.5)$ is a reasonable choice. We note that AB and AC are any $(k-1)$-frequent subsets of ABC, such that $AB \cup AC = ABC$.

Method 3. There is also an exact method to find the real support of an infrequent itemset in a database $\mathcal{D}_p$. Let $I_k = I[1].I[2] \cdots I[k]$ be a $k-$itemset consisting of k items $I[1], I[2], \cdots, I[k]$, and each item $I[i]$ in $\mathcal{D}_p$ is associated with a data structure denoted $I[i].tidlist$, which holds all the *IDs* (kept in the a sorted order) of the transactions containing $I[i]$. These tidlists are only created the first time $\mathcal{D}_p$ is scanned for frequent itemsets discovery. The support of I_k in $\mathcal{D}_p$ could be obtained using the following operation: $I_k = \frac{|\cap_{i=1}^{k} \{I[i].tidlist\}|}{|\mathcal{D}_p|}$. As we can see, this exact method comes with a space cost associated with storing the tidlists of each item in D_p.

5) CLUSTERING ASSESSMENT MEASURES

Let $C = \{C_1, C_2, \ldots, C_k\}$ be a candidate clustering of $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{n-1}\}$ generated at a similarity level $\delta \in [0, 1]$ such that $\cup_{i=1}^{k} C_i = \mathcal{D}$ and $\cap_{i=1}^{k} C_i = \emptyset$. From a graph-theoretic standpoint, each cluster C_i is a graph component in $G = (\mathcal{D}, E)$ and an edge is added between two vertices $\mathcal{D}_p$ and $\mathcal{D}_q$ if and only if $sim(\mathcal{D}_p, \mathcal{D}_q) \geq \delta$. In this subsection, we present some clustering goodness measures used to assess the quality of a candidate clustering C.

The goodness measure proposed in [22] and used for clustering evaluation in [24] is based on maximizing both the intra-cluster similarity $W(\mathcal{D})$ and the inter-cluster distance $B(\mathcal{D})$, while minimizing the number of clusters $f(\mathcal{D})$. Precisely, *goodness* measure is defined as follows:

$$goodness(\mathcal{D}) = W(\mathcal{D}) + B(\mathcal{D}) - f(\mathcal{D}) \tag{5}$$

where

$$W(\mathcal{D}) = \sum_{C_t \in C} \sum_{\mathcal{D}_i, \mathcal{D}_j \in C_t; i<j} sim(\mathcal{D}_i, \mathcal{D}_j) \times \mathbb{1}\{(\mathcal{D}_i, \mathcal{D}_j) \in E\} \tag{6}$$

and

$$B(\mathcal{D}) = \sum_{C_t, C_v \in C; t<v} \sum_{\mathcal{D}_i \in C_t, \mathcal{D}_j \in C_v; i<j} (1 - sim(\mathcal{D}_i, \mathcal{D}_j)) \tag{7}$$

such that $W(\mathcal{D})$ is a monotonic non-decreasing function on the range [0,1], that is, $W^{(i)} \leq W^{(i+1)}$ and $B(\mathcal{D})$ is a monotonic non-increasing function on [0,1], because $B^{(i)} \geq B^{(i+1)}$. The superscript i refers to the i-th generated clustering, such that at $i = 1$, each cluster is a singleton containing one database. The number of clusters $f(\mathcal{D})$ is also a monotonic non-increasing function satisfying $f^{(i)} \geq f^{(i+1)}$. Since two vertices of the same graph component might not be connected by an edge (i.e., $\{\mathcal{D}_i, \mathcal{D}_j\} \in C_k$ but $(\mathcal{D}_i, \mathcal{D}_j) \notin E$), we use the indicator term $\mathbb{1}\{(\mathcal{D}_i, \mathcal{D}_j) \in E\}$ to consider only vertices connected by an edge in $G = (\mathcal{D}, E)$. This is because each goodness measure is evaluated on a clustering generated at a given similarity level δ and if there is no edge between $\mathcal{D}_i$ and $\mathcal{D}_j$, the similarity between $\mathcal{D}_i$ and $\mathcal{D}_j$ should be less than δ, and therefore $sim(\mathcal{D}_i, \mathcal{D}_j)$ should be discarded. The candidate clustering that has the maximum value of $goodness(\mathcal{D})$ is selected as the optimal clustering.

Another goodness measure $goodness^2(\mathcal{D})$ was proposed in [25]:

$$goodness^2(\mathcal{D}) = \frac{sum\text{-}dist(\mathcal{D})}{(n^2-n)/2} + \frac{coupling(\mathcal{D})}{(n^2-n)/2} + \frac{f(\mathcal{D})-1}{n-1} \tag{8}$$

where

$$sum\text{-}dist(\mathcal{D}) = \sum_{C_t \in C} \sum_{\mathcal{D}_i, \mathcal{D}_j \in C_t; i<j} (1 - sim(\mathcal{D}_i, \mathcal{D}_j)) \times \mathbb{1}\{(\mathcal{D}_i, \mathcal{D}_j) \in E\} \tag{9}$$

and

$$coupling(\mathcal{D}) = \sum_{C_t, C_v \in C; t<v} \sum_{\mathcal{D}_i \in C_t, \mathcal{D}_j \in C_v; i<j} sim(\mathcal{D}_i, \mathcal{D}_j) \tag{10}$$

We note that $sum\text{-}dist(\mathcal{D})$ (intra-cluster distance) is a monotonic non-decreasing function, whereas $coupling(\mathcal{D})$ (inter-cluster similarity) is a monotonic non-increasing function on [0,1]. The candidate clustering that has the minimum value of $goodness^2(\mathcal{D})$ is selected as the optimal clustering.

The authors in [23] have proposed another goodness measure, denoted $goodness^3(\mathcal{D})$, defined as follows:

$$goodness^3(\mathcal{D}) = \frac{intra\text{-}sim(\mathcal{D}) + inter\text{-}dist(\mathcal{D})}{f(\mathcal{D})} \tag{11}$$

where

$$intra\text{-}sim(\mathcal{D}) = \frac{1}{f(\mathcal{D})} \sum_{C_t \in C} \begin{cases} 1, & |C_t| = 1 \\ \frac{\sum_{\mathcal{D}_i, \mathcal{D}_j \in C_t} sim(\mathcal{D}_i, \mathcal{D}_j) \times \mathbb{1}\{(\mathcal{D}_i, \mathcal{D}_j) \in E\}}{(|C_t|^2 - |C_t|)/2}, & |C_t| > 1 \end{cases} \tag{12}$$

and

$$inter\text{-}dist(\mathcal{D}) = \begin{cases} 0, & f(\mathcal{D}) = 1 \\ \sum\limits_{C_t, C_v \in C} \frac{2 \times \sum_{\mathcal{D}_i \in C_t, \mathcal{D}_j \in C_v; i<j} (1 - sim(\mathcal{D}_i, \mathcal{D}_j))}{|C_t| \times |C_v| \times (f(\mathcal{D})^2 - f(\mathcal{D}))}, & f > 1 \end{cases} \tag{13}$$

Because $intra\text{-}sim(\mathcal{D})$ and $inter\text{-}dist(\mathcal{D})$ are not only dependent on the number of clusters but also on the size of each cluster, both of them are non-monotonic on the range [0,1]. We notice that for $f(\mathcal{D}) = n$, we have $intra\text{-}sim(\mathcal{D}) = 1$ and for $f(\mathcal{D}) = 1$, $goodness^3(\mathcal{D}) = intra\text{-}sim(\mathcal{D})$. The candidate clustering that gets the maximum value of $goodness^3(\mathcal{D})$ is selected as the best clustering.

We define the Silhouette coefficient $SC(\mathcal{D}) \in [-1, 1]$ proposed in [39], [63], which is used to check how appropriately the cluster labels have been assigned to the n databases:

$$SC(\mathcal{D}) = \frac{1}{n} \sum_{i=0}^{n-1} s(\mathcal{D}_i) \tag{14}$$

where

$$s(\mathcal{D}_i) = \begin{cases} \frac{b(\mathcal{D}_i) - a(\mathcal{D}_i)}{\max\{a(\mathcal{D}_i), b(\mathcal{D}_i)\}} & if\ |C_i| > 1; \\ 0, & if\ |C_i| = 1 \end{cases} \tag{15}$$

and

$$a(\mathcal{D}_i) = \frac{\sum_{\mathcal{D}_i, \mathcal{D}_j \in C_i, \mathcal{D}_i \neq \mathcal{D}_j} (1 - sim(\mathcal{D}_i, \mathcal{D}_j)) \times \mathbb{1}\{(\mathcal{D}_i, \mathcal{D}_j) \in E\}}{|C_i - 1|} \tag{16}$$

$$b(\mathcal{D}_i) = \min_{\mathcal{D}_i \notin C_j} \frac{1}{|C_j|} \sum_{\mathcal{D}_j \in C_j} (1 - sim(\mathcal{D}_i, \mathcal{D}_j)) \tag{17}$$

We note that $\mathcal{D}_i \in C_i$ and $\mathcal{D}_j \in C_j$. A large value of $SC(\mathcal{D}) \approx 1$ indicates that the databases are highly matched to their own clusters (i.e., high cohesion) and loosely matched to neighboring clusters (i.e., low coupling).

C. PROPOSED LOSS FUNCTION

In this section, we define our loss function $L(\theta)$ and show how it is optimized. Let $X^T = [sim(\mathcal{D}_0, \mathcal{D}_1), sim(\mathcal{D}_0, \mathcal{D}_2), \ldots, sim(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})]$ be the input vector of the pairwise similarities calculated between the n databases $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_2, \ldots, \mathcal{D}_{n-1}\}$ and let $\theta^T = [\theta_{0,1}, \theta_{0,2}, \ldots, \theta_{n-2,n-1}] \in \mathbb{R}^{\binom{n}{2}}$ be the model parameters we need to learn via dual gradient descent optimization algorithm. Let $\varphi(\theta_{p,q})$ be a membership function that returns 1 if $\mathcal{D}_p$ and $\mathcal{D}_q$ belong to the same cluster and 0 otherwise.

1) INTRA-CLUSTER SIMILARITY

The weighted intra-cluster similarity of a candidate clustering on $\mathcal{D}$ is defined as follows:

$$W_\theta(\mathcal{D}) = \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} sim(\mathcal{D}_p, \mathcal{D}_q) \times \varphi(\theta_{p,q}) = X^T \cdot \varphi(\theta) \tag{18}$$

where

$$\varphi(\theta_{p,q}) = \begin{cases} 1, & \text{if } \theta_{p,q} \geq 1 \\ 0, & \text{if } \theta_{p,q} < 1 \end{cases}$$

In $W_\theta(\mathcal{D})$, we only sum the similarity values between the vertices connected by an edge in $G = (\mathcal{D}, E)$. The presence of an edge between two vertices $\mathcal{D}_p$ and $\mathcal{D}_q$ is determined by the value of its corresponding weight $\theta_{p,q}$. That is, $(\mathcal{D}_p, \mathcal{D}_q) \in E$ if and only if $\theta_{p,q} \geq 1$.

2) INTER-CLUSTER DISTANCE

The weighted inter-cluster distance of a candidate clustering on $\mathcal{D}$ is defined as follows:

$$\begin{aligned} B_\theta(\mathcal{D}) &= \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} (1 - sim(\mathcal{D}_p, \mathcal{D}_q)) \times (1 - \varphi(\theta_{p,q})) \\ &= (1 - X)^T \cdot (1 - \varphi(\theta)) \end{aligned} \tag{19}$$

In $B_\theta(\mathcal{D})$, we only sum the distances between the vertices not connected by an edge in $G = (\mathcal{D}, E)$.

3) NUMBER OF CLUSTERS

Let n be the number of nodes in the graph $G = (\mathcal{D}, E)$ and $cluster(\mathcal{D}_p)$ is a function that returns the cluster label assigned to $\mathcal{D}_p$. Initially, the number of clusters is set to n, then it decrements by one after each union operation between $cluster(\mathcal{D}_p)$ and $cluster(\mathcal{D}_q)$. Therefore, in order to find the current number of clusters, we need to subtract the number of all union operations done so far from n. However, a union operation between two disconnected clusters (i.e., $cluster(\mathcal{D}_p) \neq cluster(\mathcal{D}_q)$) is performed only if the weight associated with the edge $(\mathcal{D}_p, \mathcal{D}_q)$ is greater than or equal to 1, that is if $\theta_{p,q} \geq 1$. Hence, the number of clusters in G, denoted by $f_\theta(\mathcal{D})$, is given as follows:

$$f_\theta(\mathcal{D}) = n - \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} \mathbb{1}\{cluster(\mathcal{D}_p) \neq cluster(\mathcal{D}_q)\} \times \varphi(\theta_{p,q}) \tag{20}$$

such that $\mathbb{1}\{\cdot\}$ is the indicator function indicating the truth or falsehood of the statement passed in as argument. We should note that the maximum number of union operations we can perform in G is $n-1$. Consequently, $f_\theta(\mathcal{D})$ takes values from n down to $n - (n-1) = 1$.

4) LOSS FUNCTION DEFINITION

Initially, we set the weight vector of our model equal to the $\binom{n}{2}$ pairwise similarities between the n databases $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$, that is, $\theta^T \leftarrow X^T$. We then set the number of clusters equal to the number of databases n, that is, $f_\theta(\mathcal{D}) \leftarrow n$. After the initialization, we define the minimization problem as follows:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \mathbf{arg\,min}_{\boldsymbol{\theta}} L(\theta) \\ &= \mathbf{arg\,min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - W_\theta(\mathcal{D}))^2 \\ &= \mathbf{arg\,min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - \varphi(\theta^T) \cdot X)^2 \\ &= \mathbf{arg\,min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} sim(\mathcal{D}_p, \mathcal{D}_q) \times \varphi(\theta_{p,q}))^2 \\ &\text{where } \varphi : \mathbb{R}^{\binom{n}{2}} \rightarrow \{0,1\}^{\binom{n}{2}} \mid \varphi(\theta_{p,q}) \\ &= \frac{\text{sgn}(\theta_{p,q} - 1) + 1}{2} \\ &= \begin{cases} 1, & \text{if } \theta_{p,q} \geq 1 \\ 0, & \text{if } \theta_{p,q} < 1 \end{cases} \end{aligned} \tag{21}$$

such that $\text{sgn} : \mathbb{R}^{\binom{n}{2}} \rightarrow \{-1, 1\}^{\binom{n}{2}}$ is the signum function. The loss function $L(\theta)$ depends on two monotonic functions, the number of clusters $f_\theta(\mathcal{D})$, which is a non-increasing function, and the intra-cluster similarity $W_\theta(\mathcal{D})$, which is a non-decreasing function. Ideally, our goal is to reach the critical multi-dimensional point θ where the two scalar functions $f_\theta(\mathcal{D})$ and $W_\theta(\mathcal{D})$ are the closest possible. The normalized weight $\varphi(\theta_{p,q})$ corresponding to a similarity value $sim(\mathcal{D}_p, \mathcal{D}_q)$ is either 1 when $\mathcal{D}_p$ and $\mathcal{D}_q$ are in the same cluster or 0 when they are in different clusters. We also note that $\varphi(\cdot)$ is a shifted unit step function and its graph is depicted in Fig. 1 (a). Therefore, in order to minimize $L(\theta)$, we need to maximize the intra-cluster similarity $W_\theta(\mathcal{D})$, but at the same time, we need to maintain the absolute difference $|(f_\theta(\mathcal{D}) - W_\theta(\mathcal{D})|$ as small as possible so as to avoid putting all the databases into the same cluster, which is the case when the intra-cluster similarity reaches its maximum value at $f_\theta(\mathcal{D}) = 1$.

Example 2: Let us use the proposed similarity measure given in (3) to compute the pairwise similarities between the six databases presented with their FIs in Table 2. The 6×6

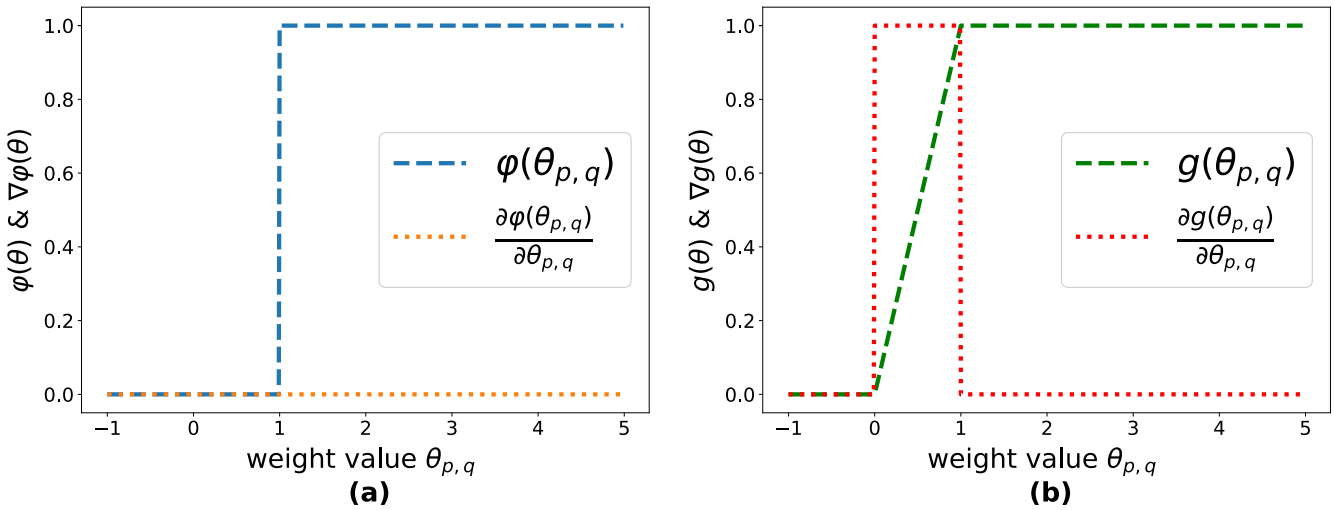


FIGURE 1. (a): represents the shifted unit step activation function $\varphi(\cdot)$ in blue and its partial derivative in orange. (b): represents the continuous piecewise linear activation function $g(\cdot)$ in green and its partial derivative in red.

similarity matrix is given in Fig. 2 (a). Then, we evaluate $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23], our proposed loss function $L(\theta)$ and the Silhouette coefficient [39] on different clusterings generated from the 6×6 similarity matrix in Fig. 2 (a) and plot their corresponding graphs in Fig. 2 (b). Due to scale differences, we multiply $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23] and Silhouette coefficient SC [39] by a factor of 2 to stretch them in the y-axis direction and we divide $L(\theta)$ by a factor of 2 to shrink it in the y-axis direction. We note that the global optimum (minimum or maximum) of each goodness evaluation measure is shown as a black point on its corresponding graph.

To generate the candidate clusterings, we use the algorithm BestDatabaseClustering [24], which proceeds as follows: initially, our similarity graph $G = (\mathcal{D}, E)$ has no edge, i.e., $|\mathcal{D}| = n$ (n=6 in this example) and $E = \emptyset$. Then, at each similarity level $\delta_i \in [0, 1]$, edges $(\mathcal{D}_p, \mathcal{D}_q)$ satisfying $sim(\mathcal{D}_p, \mathcal{D}_q) \geq \delta_i$ are added to E. The similarity level δ_i ($i = 1 \cdots m$) is selected from the m unique sorted pairwise similarity values $sim(\mathcal{D}_p, \mathcal{D}_q)$ computed between the n transaction databases, such that $p = 0 \cdots n-2$, $q = p+1 \cdots n-1$ and $\delta_1 > \delta_2 > \cdots \delta_{i-1} > \delta_i > \delta_{i+1} > \cdots > \delta_m$ and $m \leq (n^2 - n)/2$. After adding all the edges $(\mathcal{D}_p, \mathcal{D}_q)$ at δ_i, each graph component of $G = (\mathcal{D}, E)$ represents a cluster of databases in our clustering.

From Fig. 2, we notice that all the goodness measures have a global optimum at similarity level $\delta_4 = 0.36$ with $L(\theta) = 0.039$, $goodness(\mathcal{D}) = 9.97$, $goodness^2(\mathcal{D}) = 0.40$ and $goodness^3(\mathcal{D}) = 0.71$. The obtained clusters correspond exactly to the two colored regions shown in Fig. 2 (a), where the Silhouette coefficient SC=0.43 reaches its maximum value, that is, $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ and $\{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$. We also observe that our proposed loss function exhibits a quasi-convex behavior. That is, it decreases to the global minimum $L(\theta^{(4)}) = 0.039$ at $\delta_4 = 0.36$ and then increases from that point on.

Table 8 shows the different values of our loss function $L(\theta^{(i)})$ obtained on each clustering generated at a similarity level δ_i. For illustrative purposes and space limitation, we only report the weights whose values are ≥ 1 in the second column of Table 8. The weights $\theta^{(0)}_{p,q}$ are initially set to $sim(\mathcal{D}_p, \mathcal{D}_q)$, for all $1 \leq p < q \leq n$, then for each iteration $i > 0$, the weights are updated until they reach the maximum value of 1. The updating rule is given in (27). Fig. 3 depicts the graph representations of the candidate clusterings shown in Table 8.

Example 3: To illustrate the limitations of the previous goodness measures, we present another example illustrated in Fig. 4. The subfigure (a) in Fig. 4 represents a heat-map similarity matrix that comes from the experiments performed in [25], where pairwise similarities are computed between eight databases $\{D_1, D_2, \ldots, D_8\}$ generated from the synthetic dataset T10I4D100K [64]. Whereas Fig. 4 (b) represents the graphs corresponding to $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23], our proposed loss function $L(\theta)$ and the Silhouette coefficient [39] evaluated on different clusterings generated from the 8×8 similarity matrix in Fig. 4 (a).

The candidate clusterings are generated using the same process described in the previous example. Also, due to scale differences, we multiply $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23] and Silhouette coefficient SC [39] by a factor of 10 to stretch them in the y-axis direction. The use of a heat-map matrix provides a visual tool to intuitively observe the natural database clusters. A darker color represents a higher similarity value and a brighter color corresponds to a lower value. Therefore, similar databases will form homogeneous color regions when put together. Furthermore, the Silhouette plot [39] measures how appropriately the databases have been

clustered and its maximum value is used to determine the natural number of clusters.

As we notice in Fig. 4 (b), $goodness(\mathcal{D})$ has one local maximum point (13.64) at $\delta = 0.21$ with a number of clusters $f(\mathcal{D}) = 2$, and has one global maximum point (17.26) at $\delta = 0.45$, which corresponds to the maximum value of the Silhouette coefficient ($SC = 0.301$), and also matches the four clusters $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $\{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$, $\{\mathcal{D}_7\}$ and $\{\mathcal{D}_8\}$ highlighted by the four homogeneous color regions in Fig. 4 (a). The challenge we might face when using $goodness(\mathcal{D})$ is the fact that it is neither convex nor concave on the range [0,1], which makes it difficult to optimize. Furthermore, due to its non-convexity, $goodness(\mathcal{D})$ requires running over all the similarity levels δ in order to find the global maximum. This could be time consuming, especially when the number of all generated candidate clusterings is large.

On the other hand, $goodness^2(\mathcal{D})$ has one local minimum point (0.67) at $\delta = 0.45$ and $f(\mathcal{D}) = 4$, and has one global minimum point (0.58) at $\delta = 0.21$ and $f(\mathcal{D}) = 2$, which corresponds exactly to two clusters $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$, $\{\mathcal{D}_7, \mathcal{D}_8\}$. Clearly, the resulting clustering does not match the four clusters discovered by the Silhouette coefficient at $\delta = 0.45$. Also, the graph of $goodness^2(\mathcal{D})$ is neither concave nor convex on the whole interval [0,1]. Additionally, $goodness^3(\mathcal{D})$ has one global maximum point (0.56) at $\delta = 0.21$ and $f(\mathcal{D}) = 2$, which also corresponds to the two clusters $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$, $\{\mathcal{D}_7, \mathcal{D}_8\}$ that do not match the homogeneous color regions depicted in Fig. 4 (a). Now, if we examine the graph of our loss function $L(\theta)$, we notice that it has only one global minimum point (0.32) at $\delta = 0.45$ and $f_\theta(\mathcal{D}) = 4$, which corresponds exactly to the 4 clusters $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $\{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$, $\{\mathcal{D}_7\}$ and $\{\mathcal{D}_8\}$ shown Fig. 4 (a). From the previous results, we notice that using the non-convex function $goodness(\mathcal{D})$ [22] and our quasi-convex loss function $L(\theta)$, we have discovered the clustering maximizing the Silhoutte Coefficient with $SC = 0.301$ at $\delta = 0.45$. Meanwhile, using the non-convex functions $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23], we did not succeed in identifying the correct clusters.

5) OBJECTIVE OPTIMIZATION

In this subsection, we use dual gradient descent with back-propagation to find the optimal weights minimizing our loss function. Since the partial derivative of the unit step function $\varphi(\cdot)$ with respect to a weight variable $\theta_{p,q}$ is equal to zero, the weight vector θ cannot be updated during back-propagation. This is why we need to apply another differentiable activation function $g : \mathbb{R}^{\binom{n}{2}} \rightarrow [0,1]^{\binom{n}{2}}$, which is defined as follows:

$$g(\theta_{p,q}) = \max\{\theta_{p,q}, 0\} - \varphi(\theta_{p,q}) \times (\theta_{p,q} - 1) = \begin{cases} 1, & \text{if } \theta_{p,q} \geq 1 \\ \theta_{p,q}, & \text{if } \theta_{p,q} \in (0,1) \\ 0, & \text{if } \theta_{p,q} \leq 0 \end{cases}$$

$$\frac{\partial g(\theta_{p,q})}{\partial \theta_{q,q}} = \frac{\text{sgn}(\theta_{p,q}) + 1}{2} - \varphi(\theta_{p,q}) = \begin{cases} 0, & \text{if } \theta_{p,q} \geq 1 \\ 1, & \text{if } \theta_{p,q} \in (0,1) \\ 0, & \text{if } \theta_{p,q} \leq 0 \end{cases}$$

where $\frac{\partial g(\theta_{p,q})}{\partial \theta_{q,q}}$ is the partial derivative of $g(\cdot)$ with respect to the weight $\theta_{p,q}$. We also note that $g(\cdot)$ is the continuous piecewise linear activation function depicted in Fig. 1 (b). Therefore, the final constrained objective, also called the primal problem, is defined as follows:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \mathbf{arg}\,\mathbf{min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - h_\theta(\mathcal{D}))^2 \\ &= \mathbf{arg}\,\mathbf{min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - g(\theta^T) \cdot X)^2 \\ &= \mathbf{arg}\,\mathbf{min}_{\boldsymbol{\theta}} \frac{1}{2}(f_\theta(\mathcal{D}) - \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} sim(\mathcal{D}_p, \mathcal{D}_q) \times g(\theta_{p,q}))^2 \\ &\quad \text{s.t. } h_\theta(\mathcal{D}) - W_\theta(\mathcal{D}) = (g(\theta^T) - \varphi(\theta^T)) \cdot X = 0 \end{aligned} \tag{22}$$

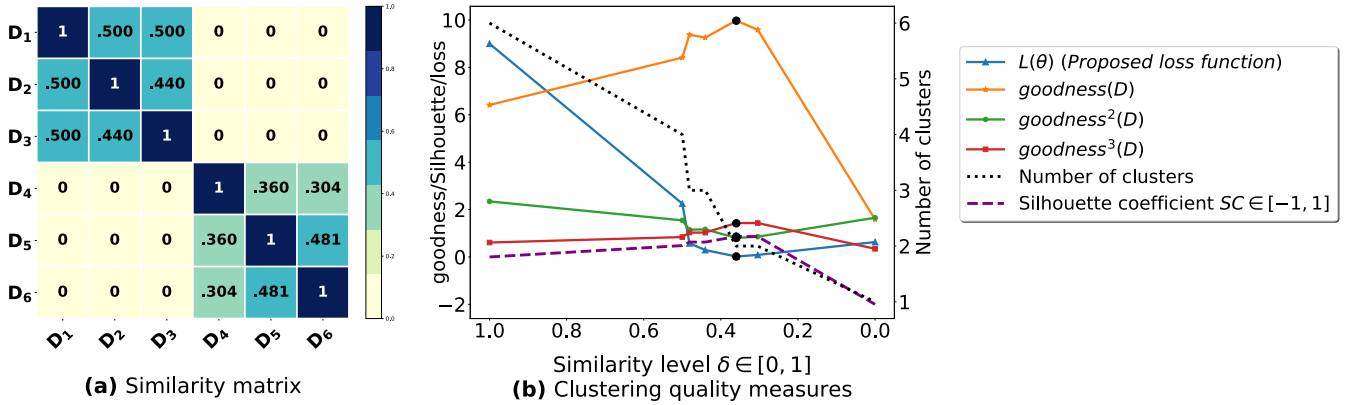
Since we have $g(\cdot) \geq \varphi(\cdot)$, the value of $h_\theta(D)$ is larger than or equal to the actual intra-cluster similarity $W_\theta(\mathcal{D})$. Consequently, $h_\theta(\mathcal{D})$ becomes rapidly larger than $f_\theta(\mathcal{D})$ due to the contribution of the weights with values $\theta_{p,q} < 1$ in calculating $h_\theta(\mathcal{D})$. This condition forces the optimization algorithm to terminate soon preventing the large weights (i.e., $1 - \theta_{p,q} \leq \epsilon | \epsilon \in]0, 1e^{-10}]$) to reach the maximum value of 1, and hence the corresponding similar database pairs $(\mathcal{D}_p, \mathcal{D}_q)$ remain in different clusters. To solve and relax this optimization problem, we need to introduce the constraint $h_\theta(\mathcal{D})$-$W_\theta(\mathcal{D}) = 0$ into the objective function using the Lagrange multiplier λ. We then use an unconstrained optimization algorithm such as dual gradient descent to find the optimal weights minimizing the objective function. Let $\mathcal{L} : \mathbb{R}^{\binom{n}{2}} \times \mathbb{R} \rightarrow \mathbb{R}$ be the Lagrangian function, which takes the constraint in the primal problem (22) and integrates it into the objective function $L(\cdot)$:

$$\begin{aligned} \mathcal{L}(\theta, \lambda) &= \frac{1}{2}(f_\theta(\mathcal{D}) - h_\theta(\mathcal{D}))^2 + \lambda(h_\theta(\mathcal{D}) - W_\theta(\mathcal{D})) \\ &= \begin{cases} L(\theta), & \text{if } h_\theta(\mathcal{D}) - W_\theta(\mathcal{D}) = 0 \\ +\infty, & \text{if } h_\theta(\mathcal{D}) - W_\theta(\mathcal{D}) \neq 0 \text{ and } \lambda \rightarrow +\infty \end{cases} \end{aligned} \tag{23}$$

The weight vector θ is referred to as primal variable, and λ as dual variable. If the constraint in (22) is satisfied, our objective function remains the same, otherwise, since we have included a non-zero term in the objective function, we need to associate a penalty to the violation of the constraint by using the scalar parameter $\lambda > 0$. Now, let us define the Lagrange dual function $\xi : \mathbb{R} \rightarrow \mathbb{R}$ over λ:

$$\xi(\lambda) = \inf_{\boldsymbol{\theta} \in \mathbb{R}^{\binom{n}{2}}} \mathcal{L}(\theta, \lambda) \tag{24}$$

where $\mathbf{inf}$ is the infimum (i.e., the greatest lower bound) over the weight vector θ. The minimum value of the Lagrange dual function is a lower bound on the optimal value of the original



(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 2. (a): 6 × 6 similarity matrix from the six databases shown in Table 1. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 2 [25], *goodness*^{3}($\mathcal{D}$) × 2 [23], the Silhouette coefficient ×2 [39], the number of clusters and our loss function $L(\theta) \div 2$.

TABLE 8. Computing our loss function $L(\theta)$ on all the candidate clusterings generated from the six databases shown in Table 1.

Similarity level δ_i	Weights $\theta^{(i)}_{p,q} \geq 1$	Edge set E_{δ_i}	# of clusters $f_{\theta^{(i)}}(\mathcal{D})$	Intra-graph similarity $W_{\theta^{(i)}}(\mathcal{D})$	Loss function $L(\theta^{(i)})$	Clusters
$0.50 < \delta_0 \leq 1$	$\emptyset$ $\forall p < q,\ \theta^{(0)}_{p,q} < 1$	$\emptyset$	$n = 6$	0	18	$\{\mathcal{D}_1\},\{\mathcal{D}_2\},\{\mathcal{D}_3\}$ $\{\mathcal{D}_4\},\{\mathcal{D}_5\},\{\mathcal{D}_6\}$
$0.481 < \delta_1 \leq 0.50$	$\theta^{(1)}_{1,2}, \theta^{(1)}_{1,3}$	$(\mathcal{D}_1,\mathcal{D}_2),(\mathcal{D}_1,\mathcal{D}_3)$	4	1.0	4.5	$\{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3\}$ $\{\mathcal{D}_4\},\{\mathcal{D}_5\},\{\mathcal{D}_6\}$
$0.44 < \delta_2 \leq 0.481$	$\theta^{(2)}_{1,2}, \theta^{(2)}_{1,3}$, $\theta^{(2)}_{5,6}$	$(\mathcal{D}_1,\mathcal{D}_2),(\mathcal{D}_1,\mathcal{D}_3)$ $(\mathcal{D}_5,\mathcal{D}_6)$	3	1.48	1.15	$\{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3\}$ $\{\mathcal{D}_4\},\{\mathcal{D}_5,\mathcal{D}_6\}$
$0.36 < \delta_3 \leq 0.44$	$\theta^{(3)}_{1,2}, \theta^{(3)}_{1,3}$, $\theta^{(3)}_{5,6}, \theta^{(3)}_{2,3}$	$(\mathcal{D}_1,\mathcal{D}_2),(\mathcal{D}_1,\mathcal{D}_3)$ $(\mathcal{D}_5,\mathcal{D}_6),(\mathcal{D}_2,\mathcal{D}_3)$	3	1.92	0.58	$\{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3\}$ $\{\mathcal{D}_4\},\{\mathcal{D}_5,\mathcal{D}_6\}$
$0.304 < \delta_4 \leq 0.36$	$\theta^{(4)}_{1,2}, \theta^{(4)}_{1,3}$, $\theta^{(4)}_{5,6}, \theta^{(4)}_{2,3}$, $\theta^{(4)}_{4,5}$	$(\mathcal{D}_1,\mathcal{D}_2),(\mathcal{D}_1,\mathcal{D}_3)$ $(\mathcal{D}_5,\mathcal{D}_6),(\mathcal{D}_2,\mathcal{D}_3)$ $(\mathcal{D}_4,\mathcal{D}_5)$	2	2.28	$L(\theta^{(4)}) = 0.039$	$\{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3\}$ $\{\mathcal{D}_4,\mathcal{D}_5,\mathcal{D}_6\}$
$0 < \delta_5 \leq 0.304$	$\theta^{(5)}_{1,2}, \theta^{(5)}_{1,3}$, $\theta^{(5)}_{5,6}, \theta^{(5)}_{2,3}$, $\theta^{(5)}_{4,5}, \theta^{(5)}_{4,6}$	$(\mathcal{D}_1,\mathcal{D}_2),(\mathcal{D}_1,\mathcal{D}_3)$ $(\mathcal{D}_5,\mathcal{D}_6),(\mathcal{D}_2,\mathcal{D}_3)$ $(\mathcal{D}_4,\mathcal{D}_5),(\mathcal{D}_4,\mathcal{D}_6)$	2	2.58	$L(\theta^{(5)}) = 0.17 > L(\theta^{(4)})$	$\{\mathcal{D}_1,\mathcal{D}_2,\mathcal{D}_3\}$ $\{\mathcal{D}_4,\mathcal{D}_5,\mathcal{D}_6\}$
$0 \leq \delta_6$	$\theta^{(6)}_{1,2},\ldots,\theta^{(6)}_{5,6}$	$(\mathcal{D}_1,\mathcal{D}_2),\ldots,(\mathcal{D}_5,\mathcal{D}_6)$	1	2.58	$L(\theta^{(6)}) = 1.25 > L(\theta^{(5)})$	$\{\mathcal{D}_1,\ldots,\mathcal{D}_6\}$

loss function (See Theorem 2). That is, if $\hat{\theta}$ is the minimum of the original loss $L(\cdot)$ then:

$$\xi(\lambda) = \inf_{\theta \in \mathbb{R}^{\binom{n}{2}}} \mathcal{L}(\theta, \lambda) \leq \mathcal{L}(\hat{\theta}, \lambda) = L(\hat{\theta}) \tag{25}$$

Therefore, solving the Lagrange dual problem corresponds to finding the best lower bound on the optimal value of the primal problem in (22). To do so, we need to solve the following maximization problem:

$$\max_{\lambda \in \mathbb{R}} \xi(\lambda) = \max_{\lambda \in \mathbb{R}} \min_{\theta \in \mathbb{R}^{\binom{n}{2}}} \mathcal{L}(\theta, \lambda) \tag{26}$$

Since $\min_{\theta} \mathcal{L}(\theta, \lambda)$ does not have a simple closed form solution, dual gradient descent is used to solve the Lagrange dual problem. In fact, as shown in [65], by performing gradient descent on the primal variable θ while doing gradient ascent on the dual variable λ, the algorithm eventually converges to the critical points of the Lagrangian $\mathcal{L}(\theta, \lambda)$. Then, by plugging back the solution $\theta^{(i)}$ into the original function $L(\cdot)$, we check if the final convergence test holds. That is, if $L(\theta^{(i)}) \leq L(\theta^{(i-1)})$, then $\theta^{(i)}$ is the local minimum solution at the iteration i. Otherwise, since $L(\cdot)$ is quasi-convex (See Theorem 3), $\theta^{(i-1)}$ is returned as the global minimizer of $L(\cdot)$.

The gradient vectors $\nabla_\theta \mathcal{L}(\theta, \lambda)$ and $\nabla_\lambda \mathcal{L}(\theta, \lambda)$ are calculated as follows:

$$\begin{aligned}
&\nabla_\theta \mathcal{L}(\theta, \lambda) \\
&= \left[\frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \theta_{0,1}}, \frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \theta_{0,2}}, \ldots, \frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \theta_{n-2,\, n-1}}\right]^{\mathrm{T}} \\
&= (-f_\theta(\mathcal{D}) + h_\theta(\mathcal{D}) + \lambda) \times X \odot (\nabla_\theta g(\theta) - \nabla_\theta \varphi(\theta)) \\
&= (-f_\theta(\mathcal{D}) + h_\theta(\mathcal{D}) + \lambda) \ \times
\end{aligned}$$

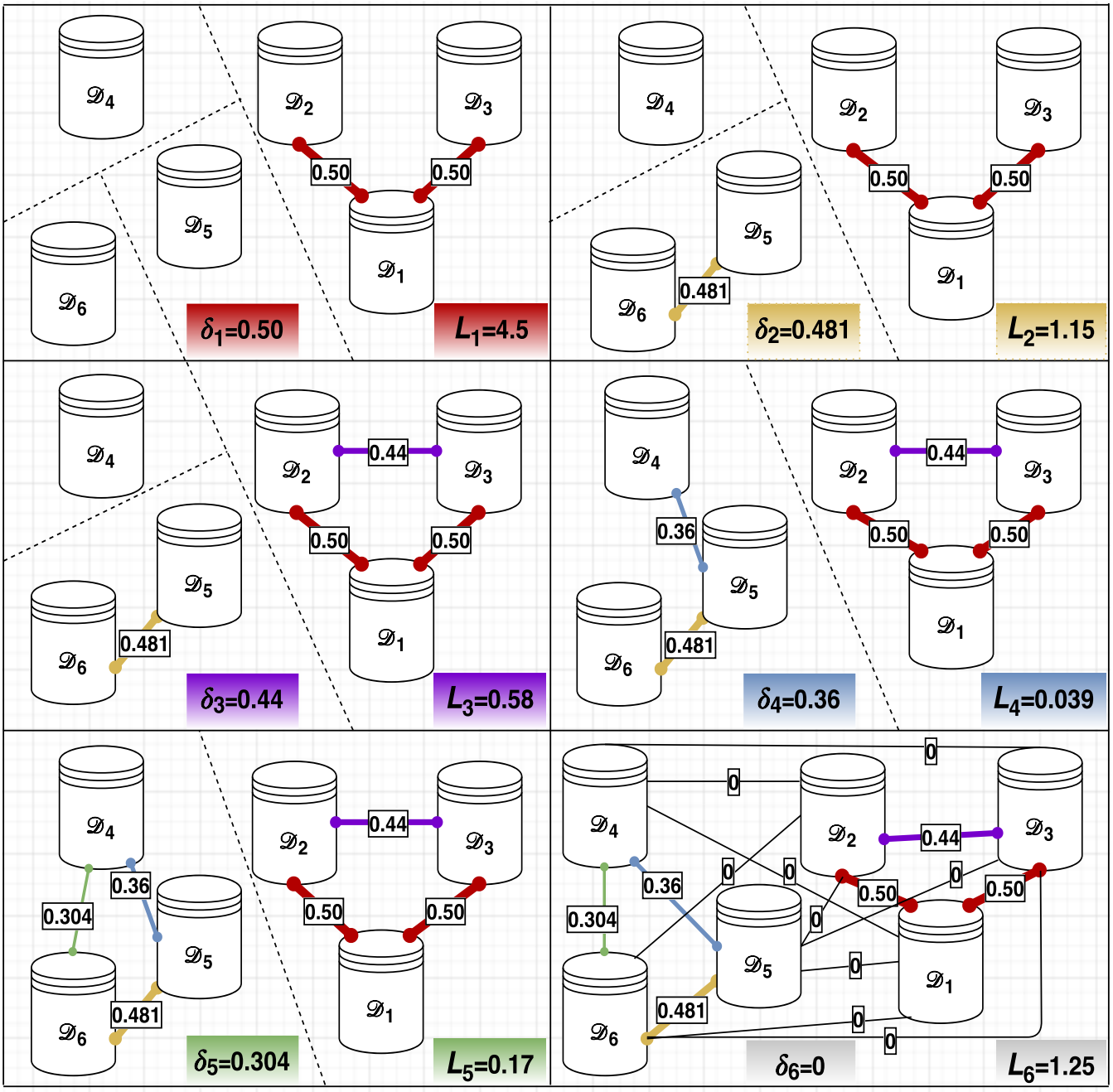


FIGURE 3. The graph representations of the candidate clusterings shown in Table 8. The figure depicts 6 candidate clusterings obtained at different similarity levels $\delta_i \in [0, 1], i = 1 \cdots 6$ and the value of our loss function $L(\theta)$ for each candidate clustering. The optimal clustering obtained at the global minimum $\min L(\theta) = 0.0039$ is $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}, \{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}$.

$$\begin{bmatrix} sim(\mathcal{D}_0, \mathcal{D}_1) \times (\frac{\partial g(\theta_{0,1})}{\partial \theta_{0,1}} - \frac{\partial \varphi(\theta_{0,1})}{\partial \theta_{0,1}}) \\ sim(\mathcal{D}_0, \mathcal{D}_2) \times (\frac{\partial g(\theta_{0,2})}{\partial \theta_{0,2}} - \frac{\partial \varphi(\theta_{0,2})}{\partial \theta_{0,2}}) \\ \vdots \\ sim(\mathcal{D}_{n-2}, \mathcal{D}_{n-1}) \times (\frac{\partial g(\theta_{n-2,n-1})}{\partial \theta_{n-2,n-1}} - \frac{\partial \varphi(\theta_{n-2,n-1})}{\partial \theta_{n-2,n-1}}) \end{bmatrix}$$

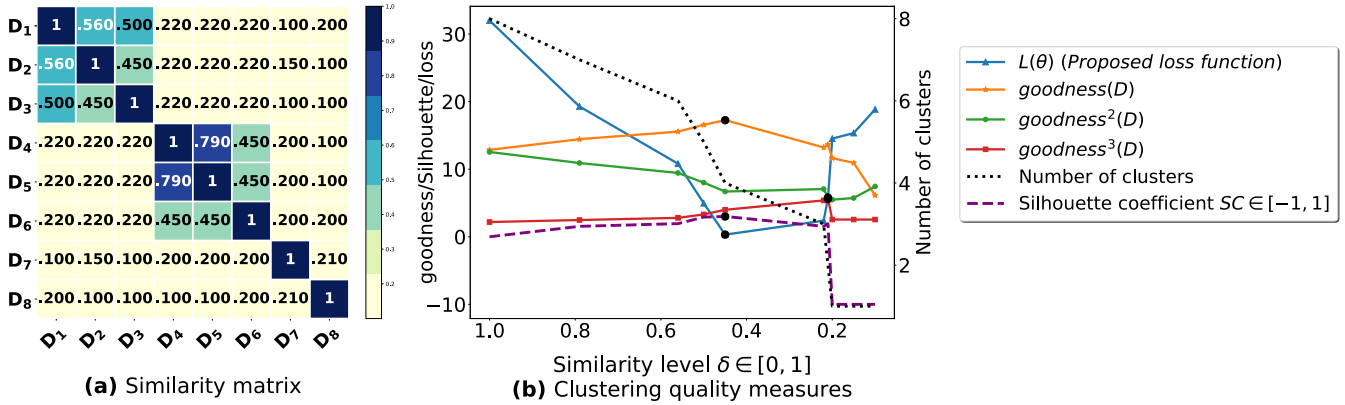
$$\nabla_\lambda \mathcal{L}(\theta, \lambda) = \frac{\partial \mathcal{L}(\theta, \lambda)}{\partial \lambda} = (h_\theta(\mathcal{D}) - W_\theta(\mathcal{D}))$$

The math operator $\odot$ represents the element-wise multiplication. Finally, given a learning rate value η, the parameters $\theta^{(i)}$ and $\lambda^{(i)}$ at the iteration i are updated iteratively as follows:

$$\theta^{(i)} = \theta^{(i-1)} - \eta(-f_{\theta^{(i-1)}}(\mathcal{D}) + h_{\theta^{(i-1)}}(\mathcal{D}) + \lambda^{(i-1)}) \times X \odot \nabla_{\theta^{(i-1)}} g(\theta^{(i-1)}) \quad (27)$$

$$\lambda^{(i)} = \lambda^{(i-1)} + \eta(h_{\theta^{(i)}}(\mathcal{D}) - W_{\theta^{(i)}}(\mathcal{D})) \quad (28)$$

As demonstrated in [65], the differential optimization algorithm will converge to a stationary point of the Lagrangian $\mathcal{L}(\theta, \lambda)$ when performing gradient descent on θ while doing



(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 4. **(a): 8 × 8 similarity matrix from the experiments performed in [25]. The pairwise similarities are computed between eight databases generated from a synthetic dataset T10I4D100K [64]. (b): represents the graphs corresponding to $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ × 10 [25], $goodness^3(\mathcal{D})$ × 10 [23], the Silhouette Coefficient SC × 10 [39], the number of generated clusters and our loss function $L(\theta)$.**

gradient ascent on λ (i.e., notice how $\lambda^{(i-1)}$ is incremented by the gradient $\nabla_{\lambda^{(i-1)}}\mathcal{L}(\theta^{(i)}, \lambda^{(i-1)})$). Therefore, we end up calculating a good local minimum of the original objective $L(\theta)$ under the constraint $h_\theta(\mathcal{D}) - W_\theta(\mathcal{D}) = 0$, and since $L(\theta)$ is quasi-convex (See Theorem 3), the obtained local minimum is nothing but the global minimizer of $L(\theta)$.

Theorem 2: Let $\hat{\theta}$ be the optimal weight vector of the primal problem defined by the original objective function $L(\cdot)$ and $\lambda \in \mathbb{R}$. The following inequality is satisfied:

$$\min_{\theta \in \mathbb{R}^{\binom{n}{2}}} \mathcal{L}(\theta, \lambda) \leq L(\hat{\theta}) \tag{29}$$

Proof: The minimum value of the Lagrangian is less than or equal to the value of the Lagrangian evaluated at another value, say $\hat{\theta}$. Therefore, we have :

$$\begin{aligned} \min \mathcal{L}(\theta, \lambda) &\leq \mathcal{L}(\hat{\theta}, \lambda) \\ &= \frac{1}{2}(f_{\hat{\theta}}(\mathcal{D}) - h_{\hat{\theta}}(\mathcal{D}))^2 - \lambda(h_{\hat{\theta}}(\mathcal{D}) - W_{\hat{\theta}}(\mathcal{D})) \quad (\to 0) \\ &= L(\hat{\theta}) \end{aligned}$$

The factor of λ is canceled out because $\hat{\theta}$ is the optimal solution of the original objective $L(\cdot)$, which means it is feasible and it verifies the equality constraint $h_{\hat{\theta}}(\mathcal{D}) = W_{\hat{\theta}}(\mathcal{D})$. Hence, by solving the Lagrangian (i.e., the relaxed problem), we obtain a lower bound on the optimal value of the original objective function $L(\cdot)$ □

6) QUASI-CONVEXITY OF THE LOSS FUNCTION

The examples presented above in Fig. 2 and Fig. 4 show that our proposed loss function exhibits a quasi-convex behavior. That is, it decreases to the global minimum and then increases from that point on. In this subsection, we attempt to prove our claim mathematically.

Theorem 3: The proposed loss function L is quasi-convex satisfying the following inequality given in [66]:

$$L(\varepsilon\theta^{(i)} + (1-\varepsilon)\theta^{(i+1)}) \leq \max\{L(\theta^{(i)}), L(\theta^{(i+1)})\} \tag{30}$$

$$\text{for all } \theta^{(i)}, \theta^{(i+1)} \in \mathbb{R}^{\binom{n}{2}} \text{ and } \varepsilon \in [0, 1] \tag{31}$$

A continuous function $L : \mathbb{R}^{\binom{n}{2}} \to \mathbb{R}$ is quasi-convex if and only if at least one of the following conditions cited in [66] holds true:

- L is non-decreasing.
- L is non-increasing.
- There is a vector $\theta^{(i)} \in \mathbb{R}^{\binom{n}{2}}$, such that for $\theta^{(i-1)} \leq \theta^{(i)}$, L is non-increasing, and for $\theta^{(i+1)} \geq \theta^{(i)}$, L is non-decreasing and $\theta^{(i)}$ is the global minimizer of L.

Proof: To demonstrate that $L(\theta)$ is quasi-convex, we need to prove the validity of (30). First, We have $f_\theta(\mathcal{D})$ is a non-increasing function on the unit interval [0,1], and hence is both quasi-convex and quasi-concave satisfying the following inequality:

$$f(\varepsilon\theta^{(i)} + (1-\varepsilon)\theta^{(i+1)}) \leq \max\{f(\theta^{(i)}), f(\theta^{(i+1)})\}$$
$$\text{for all } \theta^{(i)}, \theta^{(i+1)} \in \mathbb{R}^{\binom{n}{2}} \text{ and } \varepsilon \in [0, 1]$$

On the other hand, $W_\theta(\mathcal{D})$ is a non-decreasing function on [0,1] and is also both quasi-convex and quasi-concave satisfying the following inequality:

$$W(\varepsilon\theta^{(i)} + (1-\varepsilon)\theta^{(i+1)}) \geq \min\{W(\theta^{(i)}), W(\theta^{(i+1)})\}$$
$$\text{for all } \theta^{(i)}, \theta^{(i+1)} \in \mathbb{R}^{\binom{n}{2}} \text{ and } \varepsilon \in [0, 1]$$

By subtracting the two last inequalities, we get the following:

$$\begin{aligned} &(f(\varepsilon\theta^{(i)} + (1-\varepsilon)\theta^{(i+1)}) - W(\varepsilon\theta^{(i)} + (1-\varepsilon)\theta^{(i+1)})) \\ &\quad \leq (\max\{f(\theta^{(i)}), f(\theta^{(i+1)})\} - \min\{W(\theta^{(i)}), W(\theta^{(i+1)})\}) \end{aligned}$$

Since $f(\theta^{(i)}) \geq f(\theta^{(i+1)})$ and $W(\theta^{(i)}) \leq W(\theta^{(i+1)})$, the right side of the resulting inequality is equal to $f(\theta^{(i)}) - W(\theta^{(i)})$, which is set equal to $\max\{f(\theta^{(i)}) - W(\theta^{(i)}), f(\theta^{(i+1)}) - W(\theta^{(i+1)})\}$. Then, by squaring and dividing both sides of the inequality by two, we get a variation on Jensen's inequality for quasi-convex functions [66] as defined in (30). Therefore, our proposed loss function is quasi-convex. □

D. PROPOSED CLUSTERING MODEL

Let $G = (\mathcal{D}, E)$ be a similarity graph, such that $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{n-1}\}$ represents the vertex set of n transaction

databases and E represents the edge set, which is initially empty, $E \leftarrow \varnothing$. First, each transaction database $\mathcal{D}_p$ ($p = 0 \ldots n-1$) is mined using FP-Growth algorithm [3]. Afterward, we call our similarity function *sim* (3) on the local frequent itemsets (FIs) to compute the $\binom{n}{2}$ pairwise similarities between the n databases denoted by X^T, such that $X^T = [sim(\mathcal{D}_0, \mathcal{D}_1), sim(\mathcal{D}_0, \mathcal{D}_2), \ldots, sim(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})]$.

Each similarity value $sim(\mathcal{D}_p, \mathcal{D}_q) \in [0, 1]$ is associated with a weight $\theta_{p,q} \in \mathbb{R}$ such that similar databases will have a weight $\theta_{p,q} \geq 1$, and dissimilar databases will be associated to a weight $\theta_{p,q} \in [0, 1[$. We also note that database pairs $(\mathcal{D}_p, \mathcal{D}_p)$ with the largest weight $\theta_{p,q}$ are the first to be put into the same cluster. Let $\theta^T = [\theta_{0,1}, \theta_{0,2}, \ldots, \theta_{n-2,n-1}]$ and X^T be the parameters and the input vector fed into our perceptron-based neural network model, respectively. The proposed model is adapted to our unsupervised learning task. In fact, we do not try to learn the hyperplanes or decision boundaries to classify a training data in a supervised way (e.g., logistic regression) given their respective class labels. Instead, we use a feed-forward neural network to model the intra-cluster similarity $W_\theta(\mathcal{D})$, which is plugged into our loss function.

The simplest and straightforward way to model the similarity intra-cluster in our loss function is to use a weighted sum of the pairwise similarities, that is, $W_\theta(\mathcal{D}) = \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} sim(\mathcal{D}_p, \mathcal{D}_q) \times \varphi(\theta_{p,q})$, where each term $\varphi(\theta_{p,q}) \in \{0, 1\}$ decides whether the corresponding similarity value $sim(\mathcal{D}_p, \mathcal{D}_q)$ should take part in calculating $W_\theta(\mathcal{D})$. That is, if $\varphi(\theta_{p,q}) = 1$, an edge is added between $\mathcal{D}_p$ and $\mathcal{D}_q$ in G, $sim(\mathcal{D}_p, \mathcal{D}_q)$ is added up to $W_\theta(\mathcal{D})$, the number of clusters is decremented by one and the corresponding databases $\{\mathcal{D}_p, \mathcal{D}_q\}$ are put into the same cluster. Otherwise, if $\varphi(\theta_{p,q}) = 0$, we just keep updating the weights $\theta_{p,q}$ using gradient descent and back-propagation until convergence. Adding hidden layers between the input layer and the output unit (as it is the case in multi-layer perceptron (MLP) and deep neural networks (DNN)), will just add new weights and more complexity to our model, when we can simply model our loss function using one input layer, consisting of the pairwise similarities, and one output unit computing $W_\theta(\mathcal{D})$.

After obtaining the $\binom{n}{2}$ similarity values and initializing the learning rate η, our approach could be described as follows: initially, the weight vector θ is set equal to the input vector X, then we use dual gradient decent with back-propagation to train our model to find the optimal weights minimizing our quadratic quasi-convex loss function $L(\theta)$. Through each learning cycle η, the weights $\theta_{p,q}$ (i.e., the primal variables) are updated by making adjusted steps in the opposite direction of the gradient of the Lagrangian denoted by the term $-\eta \nabla_\theta \mathcal{L}(\theta, \lambda)$, while the Lagrange multiplier λ (i.e., dual variable) is updated by moving in the direction of the gradient of the Lagrangian denoted by the term $+\eta \nabla_\lambda \mathcal{L}(\theta, \lambda)$. Afterward, the weights $\theta_{p,q}$ that exceed the maximum value of one, will have their corresponding database pairs $\{\mathcal{D}_p, \mathcal{D}_p\}$ put into the same cluster.

The number of clusters $f_\theta(\mathcal{D})$ is initially set to the number of databases n. Then, by introducing $f_\theta(\mathcal{D})$ into the loss function $L(\theta)$, our algorithm automatically determines the optimal number of clusters that leads to minimizing $L(\theta)$. In order to maintain and track the database clusters, their sizes and their number $f_\theta(\mathcal{D})$, we use a *disjoint-set forest* data structure [67] consisting of an array of n integers managed by two main operations *union* and *cluster*. The *union* subroutine is used to link two disjoint clusters (each cluster is represented by a tree) by making the root of the smaller tree point to the root of the larger one. On the other hand, the *cluster* function is called recursively to find the cluster label assigned to the database parameter by moving down the tree towards the root. The pseudo-codes of *cluster* and *union* are provided in Algorithm 2 and Algorithm 3, respectively.

The algorithm terminates once the global minimum of the loss function has been reached. That is, for each iteration i, if $L(\theta^{(i-1)}) \geq L(\theta^{(i)})$, the algorithm updates the weights, the number of clusters, the cluster labels and then stores a copy of the clustering found so far. Otherwise, $L(\theta^{(i-1)})$ is the actual global minimum, and therefore, the algorithm outputs the optimal clustering of the n multiple databases at iteration $i-1$. The overall time complexity of our algorithm is $O(k \times \binom{n}{2})$, such that k is the number of iterations before convergence and $\binom{n}{2} = \frac{n^2-n}{2}$ is the size of the weight vector θ^T. The proposed model is illustrated in Fig. 5.

1) LEARNING RATE TUNNING

Decaying the initial value of the learning rate is an important hyper-parameter tunning in gradient-based learning algorithms. In fact, choosing a small learning rate may prevent the neural network to learn the optimal weights, whereas a large learning rate may let gradient descent overshoot the global minimum. Therefore, it is often beneficial to decrease the learning rate over time as we approach convergence. Initially,we set the learning rate η equal to a certain design value $\eta_0 \in \{1e^{-1}, 1e^{-2}, 1e^{-3} \ldots, 1e^{-10}\}$. We then use a standard decay schedule to decrease η at each iteration i as follows:

$$\eta = \eta_0 \times \frac{1.0}{1.0 + decay\_rate \times i} \tag{32}$$

such that $decay\_rate \in [1e^{-1}, 1]$ is another constant hyper-parameter that determines the rate by which η_0 is decayed, and i is the current iteration number, also called *epoch*. A grid search is then conducted to sample a randomly set of points $(\eta_0, decay\_rate)$ within a two dimensional grid in order to explore the values returning the optimal weights.

2) EDGE INDEXING

Theorem 4: Let $(\mathcal{D}_p, \mathcal{D}_q)$ be the k-th database pair such that $p = 0$ to $n-2$, $q = p+1$ to $n-1$ and $k = 1$ to $\binom{n}{2}$. Then, the indices p and q can be obtained as follows:

$$p = n - 1 - \epsilon \tag{33}$$

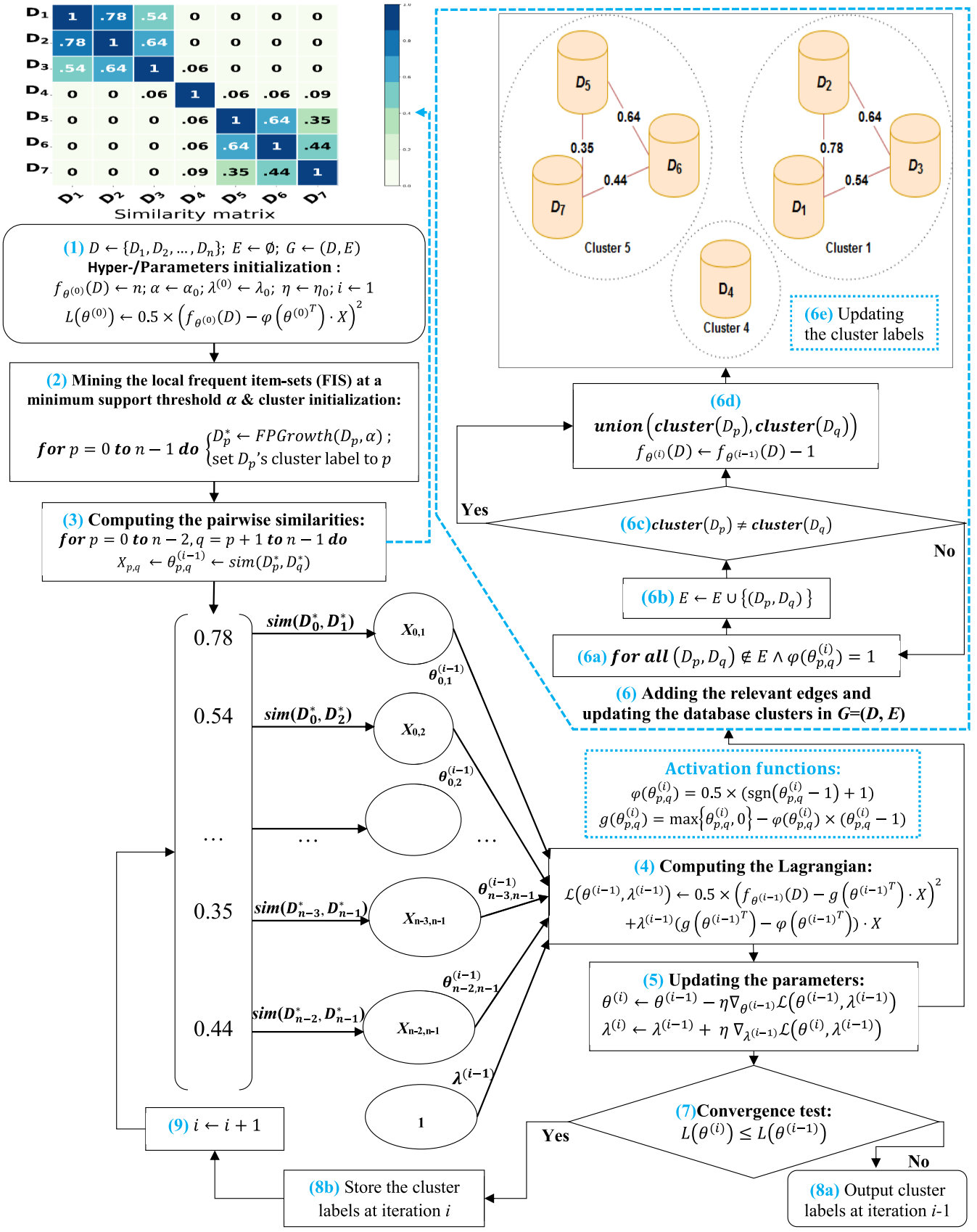


FIGURE 5. The proposed clustering model illustrated in steps numbered from (1) to (9). After step (3), the algorithm iterates over the steps (4)-(7). If the convergence test at step (7) is satisfied, the procedure terminates at step (8a), otherwise, the loop continues after steps (8b) and (9).

$$q = (p+1) + \epsilon(\epsilon+1)/2 - \left(\binom{n}{2} - k + 1\right) \quad (34)$$

where

$$\epsilon = \left\lceil \frac{\left(\sqrt{8 \times \left(\binom{n}{2} - k + 1\right) + 1} - 1\right)}{2} \right\rceil, \binom{n}{2} = \frac{n^2 - n}{2}$$

The operator $\lceil \cdot \rceil$ returns the least integer greater than or equal to the argument.

Proof: Let M be the $n \times n$ similarity table built using the similarity measure (3). Because M is a symmetric matrix, we would just consider the entries above the main diagonal. Precisely, there are $\binom{n}{2}$ similarity values associated with $\binom{n}{2}$ edges indexed from 1 to $\binom{n}{2}$.The 1^{st} row of M is associated with $n-1$ edges $(\mathcal{D}_0, \mathcal{D}_1), (\mathcal{D}_0, \mathcal{D}_2), \ldots, (\mathcal{D}_0, \mathcal{D}_{n-1})$, the 2^{nd} row with $n-2$ edges $(\mathcal{D}_1, \mathcal{D}_2), (\mathcal{D}_1, \mathcal{D}_3), \ldots, (\mathcal{D}_1, \mathcal{D}_{n-1})$, and so on until the last row, the $(n-1)^{th}$ row, which is associated with one edge $(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})$. Thus, the last ϵ rows in M are associated with $\epsilon(\epsilon+1)/2$ edges. For a given edge index k, there are $\binom{n}{2} - k + 1$ edges counting from the k^{th} edge to the last edge, the $\binom{n}{2}^{th}$ edge. To find the row index p of the k^{th} edge, we have to subtract ϵ from $n-1$, such that ϵ is the integer value that satisfies the following: $\epsilon(\epsilon+1)/2 \geq \binom{n}{2} - k + 1$. By solving the inequality we find: $(\epsilon + 1/2)^2 \geq (8\left(\binom{n}{2} - k + 1\right) + 1)/4$, $\epsilon \geq \left(\sqrt{8 \times \left(\binom{n}{2} - k + 1\right) + 1} - 1\right)/2$. Therefore, $p = n - 1 - \epsilon$ such that $\epsilon = \left\lceil \left(\sqrt{8 \times \left(\binom{n}{2} - k + 1\right) + 1} - 1\right)/2 \right\rceil$. The column index of the first edge in row p is always $p+1$. Hence, to find the column index q of the k^{th} edge, we have to add the offset $(\epsilon(\epsilon+1)/2 - \left(\binom{n}{2} - k + 1\right))$ to the column index $p+1$. □

The proposed edge indexing is an index to vertices mapping that reduces the space required to store the n^2 similarity matrix by more than half. Therefore, instead of storing the $\binom{n}{2}$ similarity values in a contiguous symmetric table of size n^2, we just need to allocate a one-dimensional array of $\binom{n}{2}$ floats in the main memory. Then, for each index $k = 1 \ldots \binom{n}{2}$, we can use equations (33) and (34) to retrieve the corresponding end-vertices $(\mathcal{D}_p, \mathcal{D}_q)$ in a constant time $O(1)$. It is worth noting that all the symmetric similarity matrices depicted in this article are only shown for illustrative purposes. In practice, they are represented by $\binom{n}{2}$ float arrays in the main memory. Assuming we have a large number of objects to cluster, say $n = 10,000$. The naive approach to storing a n^2 similarity matrix requires n^2 floats$=10,000^2 \times 8$ bytes$\approx$762.93 megabytes. Although there is no useful information in capturing the main diagonal (i.e., $sim(\mathcal{D}_i, \mathcal{D}_i) = 1$) and the lower off-main diagonal entries (i.e., $sim(\mathcal{D}_j, \mathcal{D}_i) = sim(\mathcal{D}_i, \mathcal{D}_j)$), the traditional approach allocates 64-bit for every single similarity value. Now, storing only the upper off-main diagonal values requires $\binom{n}{2}$ floats$\approx$381.43 megabytes. That is a 381.50 megabytes savings, with nearly no time overhead.

E. DATA STRUCTURE AND ALGORITHMS

In this section, we present and analyze the pseudo-code of our gradient-based clustering algorithm for multi-database mining shown in Algorithm 1: GDMDBClustering, which uses the two subroutines given in Algorithm 2: *cluster* and Algorithm 3: *union*. The proposed algorithm follows an agglomerative approach, where it starts with n singleton clusters, then database pairs are gradually merged until convergence to the global minimum of the loss function $L(\theta)$. We use the notation superscript i enclosed in round brackets, i.e., $\theta^{(i)}_{p,q}$, to indicate the iteration at which a variable $\theta_{p,q}$ has been assigned a value. Our algorithm takes the number of learning cycles, the initial leaning rate η_0 and the $\binom{n}{2}$ pairwise similarities $X^T = [sim(\mathcal{D}_0, \mathcal{D}_1), sim(\mathcal{D}_0, \mathcal{D}_2), \ldots, sim(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})]$ as arguments, and then outputs the optimal clustering at which the loss function $L(\theta)$ reaches the global minimum.

Lines 2 to 10 initialize the model parameters and the variables used in our algorithm. At line 2, the weight vector $\theta^T = [\theta_{0,1}, \theta_{0,2}, \ldots, \theta_{n-2,n-1}]$ is set to X^T. The time complexity of line 2 is $O(\binom{n}{2})$. At line 3, we initialize the disjoint-set forest data structure [67] used to maintain the cluster labels assigned to the n multiple databases. It consists of an array A of n integers managed by two operations *union* and *cluster*. Initially, we set A[p] to (-1) for $p = 0$ to $n-1$. That is, each database $\mathcal{D}_p$ is in the cluster C_p of size |A[p]|=1 (i.e., singleton), and p is the root node of the tree representing C_p in the disjoint-set data structure A. The time complexity of line 3 is $O(n)$.

At line 4, we use an array B to record the weights $\theta_{p,q}$ that have already exceeded the maximum unit value, so as to avoid checking their corresponding databases $(\mathcal{D}_p, \mathcal{D}_q)$ later in the algorithm. The time complexity of line 4 is $O(\binom{n}{2})$. At lines 5-7, we initialize the rest of the variables. That is, we set the current number of clusters $f_{\theta^{(0)}}(\mathcal{D})$ to n, whereas the Lagrange multiplier $\lambda^{(0)}$ is set to the root of the Lagrangian, and the intra-graph similarity $W_{\theta^{(0)}}(\mathcal{D})$ is set to zero. The time complexity of each line of code from 5 to 7 is constant $O(1)$. At line 8, we calculate the hypothesis function $h_{\theta^{(0)}}(\mathcal{D})$, which is defined as the dot product of the input X^T and the normalized weight vector $g(\theta^{(0)})$. The time complexity of line 8 is $O(\binom{n}{2})$. At lines 9 and 10, we calculate the loss function $L(\theta^{(0)})$ and initialize the iteration number, respectively.

At line 11, the while-loop iterates over the model parameters (i.e., the weight vector $\theta^{(i)}$ and the Lagrange multiplier $\lambda^{(i)}$) and the cluster labels to make the required updates. In the worst case, the loop terminates once the maximum number of iteration has been reached or once the number of clusters $f_{\theta^{(i)}}(\mathcal{D})$ becomes 1. At line 12, we update the learning rate. At lines 13 and 14, the weight vector $\theta^{(i)}$ and the Lagrange multiplier $\lambda^{(i)}$ are updated using the equations (27) and (28) respectively. The time complexity of lines 13 and 14 is dominated by the size of the weight vector, which is estimated to be $O(\binom{n}{2})$. At line 15, we get the indices of the weights that have just reached the maximum unit value and store them into *index_vec*. This line takes $O(\binom{n}{2})$ time. At lines 16-

Algorithm 1: GDMDBClustering

Data: n: number of databases, $X^T = [sim(\mathcal{D}_0, \mathcal{D}_1), sim(\mathcal{D}_0, \mathcal{D}_2), \ldots, sim(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})]$: pairwise similarity values
Hyper-parameters: *epochs*: number of learning cycles, η_0: initial learning rate, *decay_rate*: decay rate
Result: $\mathrm{A}_{best}[0 \ldots n-1]$: cluster labels assigned to the n databases, $f_{\theta^{(i)}}(\mathcal{D})$: number of clusters.

1 **begin**
2 $\quad\theta^{(0)} \leftarrow X$ ▷ Initialize the weight parameters
3 $\quad$**for** $p \leftarrow 0$ **to** $n-1$ **do** $\mathrm{A}[p] \leftarrow -1$ ▷ Initialize the labels assigned to the n databases
4 $\quad$**for** $k \leftarrow 0$ **to** $\binom{n}{2} - 1$ **do** $\mathrm{B}[k] \leftarrow$ **false** ▷ Vector B records which edges have already been processed
5 $\quad f_{\theta^{(0)}}(\mathcal{D}) \leftarrow n$ ▷ Initialize the number of clusters
6 $\quad\lambda^{(0)} \leftarrow -(f_{\theta^{(0)}}(\mathcal{D}) - h_{\theta^{(0)}}(\mathcal{D}))$ ▷ Initialize the Lagrange multiplier
7 $\quad W_{\theta^{(0)}}(\mathcal{D}) \leftarrow 0$ ▷ Initialize the intra-graph similarity at iteration 0
8 $\quad h_{\theta^{(0)}}(\mathcal{D}) \leftarrow X^T \cdot g(\theta^{(0)})$ ▷ Compute the hypothesis function at iteration 0
9 $\quad L(\theta^{(0)}) \leftarrow (0.5) \times (f_{\theta^{(0)}}(\mathcal{D}) - W_{\theta^{(0)}}(\mathcal{D}))^2$ ▷ Compute the loss function at iteration 0
10 $\quad i \leftarrow 1$ ▷ Initialize the iteration counter
11 $\quad$**while** $i < epochs \wedge f_{\theta^{(i)}}(\mathcal{D}) > 1$ **do**
12 $\quad\quad\eta \leftarrow \eta_0 \div (1 + decay\_rate \times i)$ ▷ Update the learning rate
13 $\quad\quad\theta^{(i)} \leftarrow \theta^{(i-1)} - \eta \nabla_{\theta^{(i-1)}} \mathcal{L}(\theta^{(i-1)}, \lambda^{(i-1)})$ ▷ Update the weight vector using (27)
14 $\quad\quad\lambda^{(i)} \leftarrow \lambda^{(i-1)} + \eta \nabla_{\lambda^{(i-1)}} \mathcal{L}(\theta^{(i)}, \lambda^{(i-1)})$ ▷ Update the Lagrange multiplier using (28)
15 $\quad\quad index\_vec \leftarrow \mathbf{arg\,where_k}\ (\varphi(\theta^{(i)}) = 1 \wedge \mathrm{B} = \mathbf{false})$ ▷ Get the edge indices with weights ≥ 1
16 $\quad\quad$**for** $k \in index\_vec$ **do**
17 $\quad\quad\quad$Let $(\mathcal{D}_p, \mathcal{D}_q)$ be the k-th database pair ▷ Vertices $(\mathcal{D}_p, \mathcal{D}_q)$ obtained by equations (33) and (34)
18 $\quad\quad\quad\mathrm{B}[k] \leftarrow$ **true** ▷ Mark the current edge as already processed
19 $\quad\quad\quad C_p \leftarrow$ ***cluster***(p, A) ▷ Get the cluster label assigned to $\mathcal{D}_p$
20 $\quad\quad\quad C_q \leftarrow$ ***cluster***(q, A) ▷ Get the cluster label assigned to $\mathcal{D}_q$
21 $\quad\quad\quad$**if** $C_p \neq C_q$ **then**
22 $\quad\quad\quad\quad$***union***(C_p, C_q, A) ▷ Cluster merging: link the smaller cluster to the larger one
23 $\quad\quad\quad\quad f_{\theta^{(i)}}(\mathcal{D}) \leftarrow f_{\theta^{(i-1)}}(\mathcal{D}) - 1$ ▷ Decrement the number of clusters after a *union* operation
24 $\quad\quad W_{\theta^{(i)}}(\mathcal{D}) \leftarrow X^T \cdot \varphi(\theta^{(i)})$ ▷ Compute the intra-graph similarity at iteration i
25 $\quad\quad h_{\theta^{(i)}}(\mathcal{D}) \leftarrow X^T \cdot g(\theta^{(i)})$ ▷ Compute the hypothesis function at iteration i
26 $\quad\quad L(\theta^{(i)}) \leftarrow (0.5) \times (f_{\theta^{(i)}}(\mathcal{D}) - W_{\theta^{(i)}}(\mathcal{D}))^2$ ▷ Compute the loss function at iteration i
$\quad\quad$▷ Test the criterion of optimality
27 $\quad\quad$**if** $L(\theta^{(i)}) \leq L(\theta^{(i-1)})$ **then**
28 $\quad\quad\quad\mathrm{A}_{best}[0 \ldots n-1] \leftarrow \mathrm{A}[0 \ldots n-1]$ ▷ Store the current clustering
29 $\quad\quad$**else**
30 $\quad\quad\quad$**break while loop** ▷ Algorithm terminates after reaching the global minimum
31 $\quad\quad i \leftarrow i + 1$
32 $\quad$**return**$(\mathrm{A}_{best}, f_{\theta^{(i-1)}}(\mathcal{D}))$

23, the for-loop examines each index k in *index_vec* to get the cluster labels and update the number of clusters $f_{\theta^{(i)}}(\mathcal{D})$ accordingly.

At line 17, we use the equations (33) and (34) to determine the database pair ($\mathcal{D}_p$,$\mathcal{D}_q$) located at the index k. Then, at line 18, the corresponding k-th weight is marked as already processed. At lines 19 and 20, the function *cluster* is called on both $\mathcal{D}_p$ and $\mathcal{D}_q$ to determine their corresponding cluster labels. If $\mathrm{A}[p] < 0$, then p the root node of the tree representing the cluster C_p, and *cluster*(p,A) returns p as the cluster label assigned to $\mathcal{D}_p$, such that $|\mathrm{A}[p]|$ is the size of C_p (i.e., the number of databases in C_p). Otherwise, if $\mathrm{A}[p] \geq 0$, then a recursive call of *cluster*(p,A) returns the cluster label assigned to $\mathcal{D}_p$ by moving down the tree from the node p towards the root node.

Once we have the two cluster labels, we test whether the two databases $\mathcal{D}_p$ and $\mathcal{D}_q$ belong to different clusters at line 21. If *cluster*(p,A) is equal to *cluster*(q,A), then $\mathcal{D}_p$ and $\mathcal{D}_q$ are already in the same cluster and therefore, no change occurs. Otherwise, a *union* operation is performed to link the root of the smaller tree (i.e., smaller cluster) to the root of the larger one (i.e., larger cluster) at line 22.

After each *union* operation, the size of the resulting cluster is set to the sum of the two cluster sizes and the number of clusters $f_{\theta^{(i)}}(\mathcal{D})$ gets decremented by one at line 23. The time complexity of a *cluster* operation is estimated to be $O(\alpha(n))$,

Algorithm 2: *cluster*

Data: A[0..n-1]: Array representing the n cluster labels
p: the index of $\mathcal{D}_p$ in A[0..n-1]
Result: The cluster label of $\mathcal{D}_p$

```
1  begin
2      if A[p] ≥ 0 then
           ▷ If p is not a root node then move down the tree towards the root
3          A[p] ←cluster(A[p], A)
4          return A[p]
5      end
6      else
7          return p ▷ return the root
8
9      end
10 end
```

Algorithm 3: *union*

Data: A[0..n-1]: Array representing the n cluster labels
C_p, C_q: Two cluster labels (*root nodes*)
Result: Applies an updating operation on array A

```
1  begin
2      if |A[C_p]| > |A[C_q]| then
3          A[C_p] ← A[C_p] + A[C_q] ▷ sum up the two cluster sizes
4          A[C_q] ← C_p ▷ establish a link to the root of the larger cluster
5      end
6      else
7          A[C_q] ← A[C_q] + A[C_p]
8          A[C_p] ← C_q
9      end
10 end
```

where $\alpha(n)$ is a function that grows very slowly, also known as the inverse of the Ackermann function [67]. In practice, $\alpha(n) \leq 4$, and we have $\alpha(n) > 4$ for very large n values $\gg 10^{80}$ as demonstrated in the chapter *Data Structures for Disjoint Sets* [67]. In the other hand, the time complexity of a *union* operation is constant $O(1)$. Therefore, the running time of the for-loop at line 16 is $O(\binom{n}{2})$.

At lines 24 and 25, the cost of calculating the intra-cluster similarity $W_{\theta^{(i)}}(\mathcal{D})$ and the hypothesis function $h_{\theta^{(i)}}(\mathcal{D})$ is $O(\binom{n}{2})$. At line 26, we calculate the loss function $L(\theta^{(i)})$ at the i-th iteration in a constant time $O(1)$. At lines 27-30, we check the condition of optimality and see whether the global minimum of the cost function has been reached. That is, for each iteration i, if $L(\theta^{(i)}) \leq L(\theta^{(i-1)})$, the algorithm stores a copy of the optimal clustering found so far and performs the next iteration. Otherwise, $L(\theta^{(i-1)})$ is the global minimum, and consequently, the algorithm returns the final clustering of the n multiple databases at iteration $i-1$. The time complexity of lines 27-30 is influenced by the operation of copying the n cluster labels. However, we can easily manage to delete this step by adding $nlog_2(n)$ extra bits to the disjoint-set forest data structure. This will help to store a copy of an individual cluster label before updating it. Therefore, the overall time complexity of our algorithm is $O(k(\binom{n}{2}))$ such that k is the number of iterations before convergence and $(\binom{n}{2})$ is the size of the weight vector θ^T.

IV. RESULTS AND DISCUSSION

In this section we present our experimental setup, data preprocessing step and discuss the obtained results in details.

A. DATA PREPARATION AND EXPERIMENTAL SETUP

To assess the performance of our algorithm in terms of accuracy and running time, we have conducted various experiments on public and synthetic data samples generated for clustering. We carried out the experiments on real world datasets, including Mushroom, Zoo and Iris, available for download from the UCI Machine Learning Repository [68], and we used a synthetic dataset T10I4D100K available on the Frequent Itemset Mining Dataset Repository [64]. These datasets are heterogeneous in terms of size (i.e., number of rows), number of clusters and number of rows per cluster. For multi-database mining scenario, we divided each dataset horizontally into $n \in \{4, 6, 10, 12\}$ partitions denoted $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$. Then, by varying the minimum support threshold $\alpha \in \{0.03, 0.2, 0.5\}$, we got different sets of frequent itemsets (FIs) using FP-Growth algorithm [3]. The details of the partitions and the FIs mined from each partition are presented in Table 9.

Once the data-preparation is done, we use our similarity measure *sim* (3) on the pairs of database partitions from each dataset to compute the similarity matrices we need for our experiments. Afterward, we cluster the database partitions using our proposed algorithm equipped with our loss function $L(\theta)$ and BestDatabaseClustering [24] equipped with one of the three goodness measures, namely $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23]. We chose the algorithm BestDatabaseClustering [24] as a baseline for our experiments because it is the fastest in practice as it was proven by the theoretical and experimental results obtained by the authors in [24]. For visualization purposes and due to scale differences, we multiply both of $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23] by a factor of 10 to stretch them in the y-axis direction and sometimes divide $L(\theta)$ by a factor of 10 or 2 to shrink it in the y-axis direction. The obtained results in terms of clustering, synthesized FIs from each cluster and running time are used for our evaluation and comparative study.

We present all the experimental results in Fig. 6 to Fig. 12 with all the details summarized in Table 10, where $\delta \in [0, 1]$ in Table 10 denotes the optimal similarity level at which each clustering quality measure reaches its optimal value (maximum or minimum). Additionally, $\eta_0 = 0.001$ is the learning rate specified at the start of the training, and $max\_epochs =$

TABLE 9. Details of the datasets, their partitions and the frequent itemsets (FIs) distribution per partition and per cluster under a minimum support threshold α.

Dataset $\mathcal{D}$	# of rows	# of rows per partition	# of $FIS(\mathcal{D}_i, \alpha)$ per partition	# of $FIS(\mathcal{D}, \alpha)$ per dataset	Ground truth clustering	# of $FIS(C_i, \alpha)$ per cluster/class
Mushroom [68] (2 clusters)	8124	$\lvert\mathcal{D}_1\rvert = 3916$ rows$(Class = 1)$ $\lvert\mathcal{D}_2\rvert = 1402$ rows$(Class = 2)$ $\lvert\mathcal{D}_3\rvert = 1402$ rows$(Class = 2)$ $\lvert\mathcal{D}_4\rvert = 1404$ rows$(Class = 2)$	$\lvert FIS(\mathcal{D}_1, 0.5)\rvert = 375$ $\lvert FIS(\mathcal{D}_2, 0.5)\rvert = 2063$ $\lvert FIS(\mathcal{D}_3, 0.5)\rvert = 32911$ $\lvert FIS(\mathcal{D}_4, 0.5)\rvert = 807$	$\lvert FIS(\mathcal{D}, 0.5)\rvert = 151$	$\boldsymbol{C_1 = \{\mathcal{D}_1\}}$, $\boldsymbol{C_2 = \{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}}$	$\lvert FIS(C_1, 0.5)\rvert = 375$ $\lvert FIS(C_2, 0.5)\rvert = 1441$
Zoo [68] (7 clusters)	101	$\lvert\mathcal{D}_1\rvert = 20$ rows$(Class = 1)$ $\lvert\mathcal{D}_2\rvert = 21$ rows$(Class = 1)$ $\lvert\mathcal{D}_3\rvert = 10$ rows$(Class = 2)$ $\lvert\mathcal{D}_4\rvert = 10$ rows$(Class = 2)$ $\lvert\mathcal{D}_5\rvert = 5$ rows$(Class = 3)$ $\lvert\mathcal{D}_6\rvert = 6$ rows$(Class = 4)$ $\lvert\mathcal{D}_7\rvert = 7$ rows$(Class = 4)$ $\lvert\mathcal{D}_8\rvert = 2$ rows$(Class = 5)$ $\lvert\mathcal{D}_9\rvert = 2$ rows$(Class = 5)$ $\lvert\mathcal{D}_{10}\rvert = 4$ rows$(Class = 6)$ $\lvert\mathcal{D}_{11}\rvert = 4$ rows$(Class = 6)$ $\lvert\mathcal{D}_{12}\rvert = 10$ rows$(Class = 7)$	$\lvert FIS(\mathcal{D}_1, 0.5)\rvert = 24383$ $\lvert FIS(\mathcal{D}_2, 0.5\rvert = 30975$ $\lvert FIS(\mathcal{D}_3, 0.5)\rvert = 30719$ $\lvert FIS(\mathcal{D}_4, 0.5)\rvert = 32767$ $\lvert FIS(\mathcal{D}_5, 0.5)\rvert = 20479$ $\lvert FIS(\mathcal{D}_6, 0.5\rvert = 65535$ $\lvert FIS(\mathcal{D}_7, 0.5)\rvert = 65535$ $\lvert FIS(\mathcal{D}_8, 0.5)\rvert = 114687$ $\lvert FIS(\mathcal{D}_9, 0.5)\rvert = 98303$ $\lvert FIS(\mathcal{D}_{10}, 0.5)\rvert = 53247$ $\lvert FIS(\mathcal{D}_{11}, 0.5)\rvert = 57343$ $\lvert FIS(\mathcal{D}_{12}, 0.5)\rvert = 28671$	$\lvert FIS(\mathcal{D}, 0.5)\rvert = 0$	$\boldsymbol{C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}}$, $\boldsymbol{C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}}$, $\boldsymbol{C_3 = \{\mathcal{D}_5\}}$, $\boldsymbol{C_4 = \{\mathcal{D}_6, \mathcal{D}_7\}}$, $\boldsymbol{C_5 = \{\mathcal{D}_8, \mathcal{D}_9\}}$, $\boldsymbol{C_6 = \{\mathcal{D}_{10}, \mathcal{D}_{11}\}}$, $\boldsymbol{C_7 = \{\mathcal{D}_{12}\}}$,	$\lvert FIS(C_1, 0.5)\rvert = 25087$ $\lvert FIS(C_2, 0.5)\rvert = 28671$ $\lvert FIS(C_3, 0.5)\rvert = 2479$ $\lvert FIS(C_4, 0.5)\rvert = 49151$ $\lvert FIS(C_5, 0.5)\rvert = 57343$ $\lvert FIS(C_6, 0.5)\rvert = 45055$ $\lvert FIS(C_7, 0.5)\rvert = 28671$
Iris [68] (3 clusters)	150	$\lvert\mathcal{D}_1\rvert = 25$ rows$(Class = 1)$ $\lvert\mathcal{D}_2\rvert = 25$ rows$(Class = 1)$ $\lvert\mathcal{D}_3\rvert = 25$ rows$(Class = 2)$ $\lvert\mathcal{D}_4\rvert = 25$ rows$(Class = 2)$ $\lvert\mathcal{D}_5\rvert = 25$ rows$(Class = 3)$ $\lvert\mathcal{D}_6\rvert = 25$ rows$(Class = 3)$	$\lvert FIS(\mathcal{D}_1, 0.2)\rvert = 5$ $\lvert FIS(\mathcal{D}_2, 0.2)\rvert = 6$ $\lvert FIS(\mathcal{D}_3, 0.2)\rvert = 2$ $\lvert FIS(\mathcal{D}_4, 0.2)\rvert = 2$ $\lvert FIS(\mathcal{D}_5, 0.2)\rvert = 2$ $\lvert FIS(\mathcal{D}_6, 0.2)\rvert = 5$	$\lvert FIS(\mathcal{D}, 0.2)\rvert = 0$	$\boldsymbol{C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}}$, $\boldsymbol{C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}}$, $\boldsymbol{C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}}$	$\lvert FIS(C_1, 0.2)\rvert = 3$ $\lvert FIS(C_2, 0.2)\rvert = 1$ $\lvert FIS(C_3, 0.2)\rvert = 2$
T10I4D100K [64] (unknown clusters)	100,000	$\lvert D_i\rvert = 10,000$ rows, $i = 1 \ldots 10$	$\lvert FIS(\mathcal{D}_1, 0.03)\rvert = 58$ $\lvert FIS(\mathcal{D}_2, 0.03)\rvert = 58$ $\lvert FIS(\mathcal{D}_3, 0.03)\rvert = 62$ $\lvert FIS(\mathcal{D}_4, 0.03)\rvert = 57$ $\lvert FIS(\mathcal{D}_5, 0.03)\rvert = 62$ $\lvert FIS(\mathcal{D}_6, 0.03)\rvert = 63$ $\lvert FIS(\mathcal{D}_7, 0.03)\rvert = 63$ $\lvert FIS(\mathcal{D}_8, 0.03)\rvert = 59$ $\lvert FIS(\mathcal{D}_9, 0.03)\rvert = 61$ $\lvert FIS(\mathcal{D}_{10}, 0.03)\rvert = 62$	$\lvert FIS(\mathcal{D}, 0.03)\rvert = 50$	7 clusters found using the Silhouette Coefficient [39] $\boldsymbol{C_1 = \{\mathcal{D}_1\}}$, $\boldsymbol{C_2 = \{\mathcal{D}_2\}}$, $\boldsymbol{C_3 = \{\mathcal{D}_3\}}$, $\boldsymbol{C_4 = \{\mathcal{D}_4, \mathcal{D}_5\}}$, $\boldsymbol{C_5 = \{\mathcal{D}_6\}}$, $\boldsymbol{C_6 = \{\mathcal{D}_7\}}$, $\boldsymbol{C_7 = \{\mathcal{D}_8, \mathcal{D}_9, \mathcal{D}_{10}\}}$	$\lvert FIS(C_1, 0.03)\rvert = 58$ $\lvert FIS(C_2, 0.03)\rvert = 58$ $\lvert FIS(C_3, 0.03)\rvert = 62$ $\lvert FIS(C_4, 0.03)\rvert = 59$ $\lvert FIS(C_5, 0.03)\rvert = 63$ $\lvert FIS(C_6, 0.03)\rvert = 59$ $\lvert FIS(C_7, 0.03)\rvert = 61$
Fig. 6 [22] (unknown clusters)	24	$\lvert\mathcal{D}_1\rvert = 3$ rows $\lvert\mathcal{D}_2\rvert = 3$ rows $\lvert\mathcal{D}_3\rvert = 3$ rows $\lvert\mathcal{D}_4\rvert = 4$ rows $\lvert\mathcal{D}_5\rvert = 4$ rows $\lvert\mathcal{D}_6\rvert = 3$ rows $\lvert\mathcal{D}_7\rvert = 4$ rows	$\lvert FIS(\mathcal{D}_1, 0.42)\rvert = 3$ $\lvert FIS(\mathcal{D}_2, 0.42)\rvert = 3$ $\lvert FIS(\mathcal{D}_3, 0.42)\rvert = 5$ $\lvert FIS(\mathcal{D}_4, 0.42)\rvert = 7$ $\lvert FIS(\mathcal{D}_5, 0.42)\rvert = 7$ $\lvert FIS(\mathcal{D}_6, 0.42)\rvert = 5$ $\lvert FIS(\mathcal{D}_7, 0.42)\rvert = 3$	$\lvert FIS(\mathcal{D}, 0.42)\rvert = 0$	3 clusters found using the Silhouette Coefficient [39] $\boldsymbol{C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}}$ $\boldsymbol{C_2 = \{\mathcal{D}_4\}}$ $\boldsymbol{C_3 = \{\mathcal{D}_5, \mathcal{D}_6, \mathcal{D}_7\}}$	$\lvert FIS(C_1, 0.42)\rvert = 3$ $\lvert FIS(C_2, 0.42)\rvert = 7$ $\lvert FIS(C_3, 0.42)\rvert = 3$

1000 is the maximum number of iterations set for all the experiments. The learning rate decreases with each epoch i using (32). We should note that the global optimum (minimum or maximum) of each goodness evaluation measure is shown as a black point on its corresponding graph. In Table 9, we compare the number of frequent itemsets (FIs) mined from the whole dataset (the fifth column of Table 9) with the FIs synthesized from the individual ground truth clusters (the most right column of Table 9). The purpose is to demonstrate the importance of clustering the multiple databases into cohesive groups prior to mining the global patterns.

All the algorithms have been implemented in Python version 3.6.7 and run on Google Colaboratory, a free Jupyter notebook environment running on Ubuntu-18.04 server equipped with Intel(R) Xeon(R) CPU clocked @2.30GHz with 12.6 GB available RAM and 33 GB available Disk.

B. EXPERIMENTAL RESULTS AND DISCUSSION

1) QUASI-CONVEXITY ANALYSIS AND CLUSTERING ACCURACY

We ran the first part of our experiments on some similarity matrices computed during the data preparation step and on some matrices taken from the experiments done in [22], [23], [36] . The purpose of this study is to plot and analyze the behavior of the graphs produced by our proposed loss function $L(\theta)$ and the existing clustering evaluation measures in [22], [23], [25]. We also compare the clustering returned by our algorithm with the cluster labels generated by Best-DatabaseClustering [24]. The experimental results are shown in Fig. 6 to Fig. 12. Each figure depicts two subfigures, (a) and (b), which represent, respectively, the heat-map similarity matrix of size $n \times n$ and the graphs corresponding to our loss function $L(\theta)$, the number of generated clusters, the Silhouette coefficient *SC* [39], *goodness*$(\mathcal{D})$ [22], *goodness*$^2(\mathcal{D})$ [25] and *goodness*$^3(\mathcal{D})$ [23].

We note that, the heat-map similarity matrix provides a visual tool for external clustering assessment, such that a darker color corresponds to a higher similarity value, whereas a brighter color corresponds to a lower similarity value. Therefore, unlike dissimilar databases, similar databases tend to form homogeneous color regions representing the clusters we need to discover. When the actual number of clusters and class labels are unknown such as the case for dataset T10I4D100K [64], we use the clustering labels produced at

TABLE 10. Clustering results shown in figures Fig. 6 to Fig. 12 obtained using the different evaluation measures, $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25], $goodness^3(\mathcal{D})$ [23], the Silhouette Coefficient SC [39] and our proposed loss function $L(\theta)$, where δ is the similarity level corresponding to the global optimum of each clustering evaluation function.

Dataset $\mathcal{D}$	Silhouette Coefficient	Clustering output/ Loss function		Clustering output/ Objective		Clustering output/ Objective		Clustering output/ Objective	
	max SC	*clusters*	min $L(\theta)$	clusters	max $goodness(\mathcal{D})$	clusters	min $goodness^2(\mathcal{D})$	clusters	max $goodness^3(\mathcal{D})$
Fig. 6 7×7 [22]	0.46 at $\delta = 0.444$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $C_2 = \{\mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6, \mathcal{D}_7\}$	0.00068 at $\delta = 0.444$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $C_2 = \{\mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6, \mathcal{D}_7\}$	15.407 at $\delta = 0.444$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_7\}$	0.259 at $\delta = 0.065$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $C_2 = \{\mathcal{D}_4, \ldots, \mathcal{D}_7\}$	0.728 at $\delta = 0.086$
Fig. 7 12×12 *Zoo* [68]	0.41 at $\delta = 0.559$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5\}$, $C_4 = \{\mathcal{D}_6, \mathcal{D}_7\}$, $C_5 = \{\mathcal{D}_8, \mathcal{D}_9\}$, $C_6 = \{\mathcal{D}_{10}, \mathcal{D}_{11}\}$, $C_7 = \{\mathcal{D}_{12}\}$	5.93 at $\delta = 0.559$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5\}$, $C_4 = \{\mathcal{D}_6, \mathcal{D}_7\}$, $C_5 = \{\mathcal{D}_8, \mathcal{D}_9\}$, $C_6 = \{\mathcal{D}_{10}, \mathcal{D}_{11}\}$, $C_7 = \{\mathcal{D}_{12}\}$	32.98 at $\delta = 0.559$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_{12}\}$	0.57 at $\delta = 0.348$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_{12}\}$	0.42 at $\delta = 0.348$
Fig. 8 4×4 *Mushroom* [68]	0.08 at $\delta = 0.41$	$C_1 = \{\mathcal{D}_1\}$, $C_2 = \{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$	0.27 at $\delta = 0.41$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_4\}$	1.672 at $\delta = 0.365$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_4\}$	0.55 at $\delta = 0.365$	$C_1 = \{\mathcal{D}_1\}$, $C_2 = \{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$	0.68 at $\delta = 0.41$
Fig. 9 6×6 *Iris* [68]	0.304 at $\delta = 0.3$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}$	1.81 at $\delta = 0.3$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}$	9.64 at $\delta = 0.3$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}$	0.55 at $\delta = 0.3$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}$, $C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}$	0.44 at $\delta = 0.3$
Fig. 10 6×6 *Zoo* & *Mushroom* [68]	0.63 at $\delta = 0.384$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \ldots, \mathcal{D}_6\}$	1.93 at $\delta = 0.384$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \ldots, \mathcal{D}_6\}$	9.96 at $\delta = 0.384$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \ldots, \mathcal{D}_6\}$	0.40 at $\delta = 0.384$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $C_2 = \{\mathcal{D}_3, \ldots, \mathcal{D}_6\}$	0.85 at $\delta = 0.384$
Fig. 11 4×4 [36]	0.34 at $\delta = 0.429$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $C_2 = \{\mathcal{D}_4\}$	0.067 at $\delta = 0.429$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_4\}$	2.708 at $\delta = 0.25$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_4\}$	0.38 at $\delta = 0.25$	$C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$, $C_2 = \{\mathcal{D}_4\}$	0.81 at $\delta = 0.429$
Fig. 12 10×10 T10I4D100K [64]	0.115 at $\delta = 0.846$	$C_1 = \{\mathcal{D}_1\}$, $C_2 = \{\mathcal{D}_2\}$, $C_3 = \{\mathcal{D}_3\}$, $C_4 = \{\mathcal{D}_4, \mathcal{D}_5\}$, $C_5 = \{\mathcal{D}_6\}$, $C_6 = \{\mathcal{D}_7\}$, $C_7 = \{\mathcal{D}_8, \mathcal{D}_9, \mathcal{D}_{10}\}$	6.191 at $\delta = 0.846$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_{10}\}$	35.275 at $\delta = 0.737$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_{10}\}$	0.193 at $\delta = 0.737$	$C_1 = \{\mathcal{D}_1, \ldots, \mathcal{D}_{10}\}$	0.806 at $\delta = 0.737$

the maximum value of the Silhouette coefficient SC [39] as a ground truth indicator for comparison.

Let us start our experiments by considering the same dataset example presented in [22]. Let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_7\}$ be a set of seven transaction databases, where each database has set of transactions enclosed in parentheses and each transaction has some items separated by commas as shown in Table 11.

The number of transactions corresponding to the seven databases are 3, 3, 3, 4, 4, 3 and 4, respectively. Let $\alpha = 0.42$ be the minimum support threshold set by the user. The FIs mined in each local database are given in Table 12. If we use the equation given in (1) to compute the global support of each itemset $I_k \in \cup_{p=1}^{7}\{FIS(\mathcal{D}_p, 0.42)\}$ in the whole dataset $\mathcal{D}$, we notice that all the synthesized global itemsets are infrequent (i.e., $\forall\ I_k \in FIS(\mathcal{D}, 0.42), supp(I_k, \mathcal{D}) < \alpha = 0.42$). Therefore, no novel pattern has been found. This is because irrelevant databases took part in the synthesizing process. Now, let us cluster the seven databases using our algorithm and BestDatabaseClustering [24] equipped with the goodness measures $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23]. The obtained results are shown in Fig. 6 and Table 10.

From Fig. 6 (b) and the first row of Table 10, we can see that using our loss function $L(\theta)$ and $goodness(\mathcal{D})$ [22], we succeeded in finding the optimal clustering, $C_1 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}, C_2 = \{\mathcal{D}_4\}, C_3 = \{\mathcal{D}_5, \mathcal{D}_6, \mathcal{D}_7\}$ at $\delta = 0.44$ with a maximum Silhouette coefficient $SC(\mathcal{D}) = 0.46$. Conversely, $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23] have failed to identify the expected clustering that maximizes the Silhouette coefficient. Moreover, we notice that our loss function has a quasi-convex behavior, which allows us to end the clustering process just after the global minimum point. On the other hand, $goodness(\mathcal{D})$ [22] and $goodness^2(\mathcal{D})$ [25] have few local optimum points, which requires the algorithm to continue visiting all the similarity levels before terminating and returning the optimal clustering.

TABLE 11. Transaction databases (TD) from [22].

TD	$Transactions$
$\mathcal{D}_1$	$(A, B, C), (A, C), (A, C, D)$
$\mathcal{D}_2$	$(A, C), (A, B), (A, C, E)$
$\mathcal{D}_3$	$(A, E), (A, C, E), (A, B, C)$
$\mathcal{D}_4$	$(F, D), (F, D, H), (E, F, D), (E, F, H)$
$\mathcal{D}_5$	$(G, H, I), (I, J), (H, I), (I, J, G)$
$\mathcal{D}_6$	$(G, H, I), (I, J, H), (I, J)$
$\mathcal{D}_7$	$(A, B), (G, H), (H, I), (H, I, J)$

Now, using the same synthesizing model in (1) on each cluster separately, we discover new valid frequent itemsets with a support above the minimum support

TABLE 12. Frequent itemsets (FIs) mined from databases $\mathcal{D}_i$ in Table 11 under a minimum support threshold $\alpha = 0.42$ and for $i = 1$ to $n = 7$.

TD	$FIS(\mathcal{D}_i, \alpha)$
$\mathcal{D}_1$	$\langle A, 1\rangle, \langle AC, 1\rangle, \langle C, 1.0\rangle$
$\mathcal{D}_2$	$\langle A, 1\rangle, \langle AC, 0.66\rangle, \langle C, 0.66\rangle$
$\mathcal{D}_3$	$\langle A, 1\rangle, \langle AE, 0.66\rangle, \langle AC, 0.66\rangle, \langle E, 0.66\rangle, \langle C, 0.66\rangle$
$\mathcal{D}_4$	$\langle H, 0.5\rangle, \langle FH, 0.5\rangle, \langle F, 1\rangle, \langle E, 0.5\rangle, \langle D, 0.75\rangle, \langle EF, 0.5\rangle, \langle DF, 0.75\rangle$
$\mathcal{D}_5$	$\langle H, 0.5\rangle, \langle G, 0.5\rangle, \langle IJ, 0.5\rangle, \langle J, 0.5\rangle, \langle HI, 0.5\rangle, \langle I, 1\rangle, \langle GI, 0.5\rangle$
$\mathcal{D}_6$	$\langle J, 0.66\rangle, \langle I, 1\rangle, \langle HI, 0.66\rangle, \langle H, 0.66\rangle, \langle IJ, 0.66\rangle$
$\mathcal{D}_7$	$\langle H, 0.75\rangle, \langle HI, 0.5\rangle, \langle I, 0.5\rangle$

TABLE 13. Frequent itemsets (FIs) mined from the database partitions $\mathcal{D}_i$ of the Iris dataset under a minimum support threshold $\alpha = 0.2$ for $i = 1$ to $n = 6$.

TD	$FIS(\mathcal{D}_i, \alpha)$
$\mathcal{D}_1$	$\langle A_1 = 5.1, 0.2\rangle, \langle A_2 = 3.4, 0.2\rangle, \langle A_3 = 1.5, 0.28\rangle, \langle A_3 = 1.4, 0.28\rangle, \langle A_4 = 0.2, 0.52\rangle$
$\mathcal{D}_2$	$\langle A_1 = 5.0, 0.24\rangle, \langle A_3 = 1.6, 0.24\rangle, \langle A_3 = 1.5, 0.24\rangle, \langle A_3 = 1.4, 0.24\rangle, \langle A_3 = 1.3, 0.2\rangle, \langle A_4 = 0.2, 0.64\rangle$
$\mathcal{D}_3$	$\langle A_4 = 1.3, 0.24\rangle, \langle A_4 = 1.5, 0.28\rangle$
$\mathcal{D}_4$	$\langle A_2 = 3.0, 0.24\rangle, \langle A_4 = 1.3, 0.28\rangle$
$\mathcal{D}_5$	$\langle A_2 = 3.0, 0.2\rangle, \langle A_4 = 1.8, 0.2\rangle$
$\mathcal{D}_6$	$\langle A_2 = 2.8, 0.2\rangle, \langle A_2 = 3.0, 0.28\rangle, \langle A_3 = 5.6, 0.2\rangle, \langle A_4 = 1.8, 0.24\rangle, \langle A_4 = 2.3, 0.2\rangle$

threshold $\alpha = 0.42$. Precisely, $FIS(C_1, 0.42) = \{\langle A, 1\rangle, \langle AC, 0.77\rangle, \langle C, 0.77\rangle\}$, $FIS(C_2, 0.42) = FIS(\mathcal{D}_4, 0.42)$ and $FIS(C_3, 0.42) = \{\langle I, 0.818\rangle, \langle IH, 0.54\rangle, \langle H, 0.63\rangle\}$. In other words, more than 77% of the transactions in C_1 contain itemsets A, AC and C, and more than 54% of the transactions in C_3 contain I, IH and H. As we can see, analyzing the local frequent itemsets coming from the same cluster allowed us to discover novel and useful information for the decision making process.

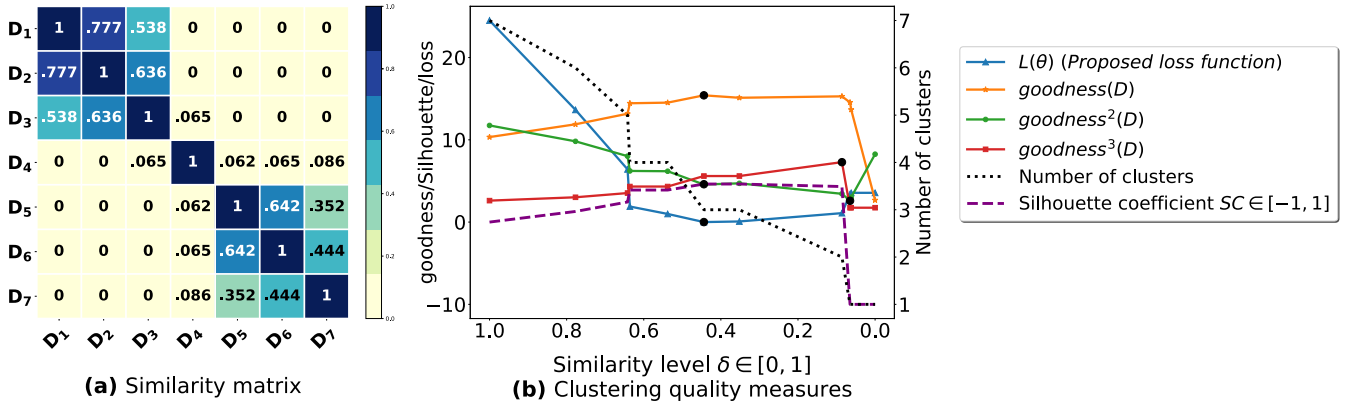
Next, we have conducted some experiments on the real-world and synthetic datasets described in Table 9. We notice in Fig. 7 (b) and Fig. 12 (b) (the second and seventh rows of Table 10) that $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23] reach their optimum values when the number of clusters $f(\mathcal{D}) = 1$, that is, when all the databases are in the same cluster. The same behavior occurs in Fig. 8 (b), Fig. 11 (b) and Fig. 12 (b) (the third, the sixth and seventh rows of Table 10) where both $goodness(\mathcal{D})$ [22] and $goodness^2(\mathcal{D})$ [25] have identified the trivial 1-class clustering as optimal. Using our loss function $L(\theta)$ instead, we were able to find the clustering maximizing the Silhouette coefficient SC. In fact, we can see that in Fig. 12 (b) and the seventh row of Table 10, our loss function was the only one to identify the optimal clustering at $\delta = 0.846$.

From the above experiments, we observe that $goodness(\mathcal{D})$ [22], $goodness^2(\mathcal{D})$ [25] and $goodness^3(\mathcal{D})$ [23] are neither quasi-convex nor quasi-concave on the whole set of the similarity domain. Consequently, we notice the presence of local optima, which makes finding the global optimum an NP-hard problem due to the unpredictable behavior of the objective. On the other hand, $L(\theta)$ is quasi-convex with no local minima. That is, for any two multi-dimensional weight vectors $\theta^{(i)}$, $\theta^{(i+1)} \in \mathbb{R}^{\binom{n}{2}}$, such that $L(\theta^{(i)}) \geq L(\theta^{(i+1)})$, the line joining the points $(\theta^{(i)}, L(\theta^{(i)}))$ and $(\theta^{(i+1)}, L(\theta^{(i)}))$ lies above the graph of the loss function $L(\theta)$, which is the case in all the figures from Fig. 6 to Fig. 12. Therefore, using $L(\theta)$, our algorithm guarantees convergence to the global minimum.

In Table 9, we show the ground-truth clustering and compare the FIs mined from a whole dataset (the fifth column of Table 9) against the FIs mined from each individual cluster of the same dataset (the most right column of Table 9). We can clearly see that mining the datasets Zoo [68] and Iris [68] under a predefined minimum support threshold α does not return any valid frequent itemset (FI). Exceptionally, in Mushroom [68], only few noisy patterns have been mined. On the other hand, synthesizing the global patterns from each individual cluster of the datasets Zoo [68] and Iris [68], results in improving the quality of the global patterns mined from each dataset, which is indicated by the number of useful patterns obtained from each cluster.

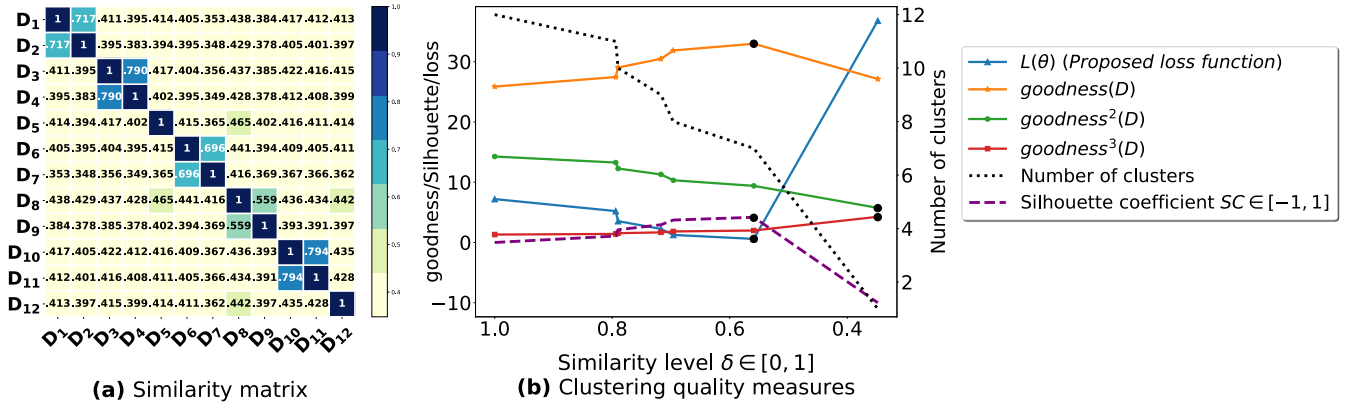
To support these observations, let us examine in details the clustering applied on the Iris dataset. Let $\mathcal{D}_{iris} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_6\}$ be the set of six database partitions obtained from the Iris dataset such that each database has 25 instances from one of the three classes C_1=Iris-Setosa, C_2=Iris-Versicolour or C_3=Iris-Virginica. The details of the 6 partitions are given in Table 9. We note that each sample in $\mathcal{D}_{iris}$ has 4 attributes, namely A_1: "Sepal length in cm", A_2: "Sepal width in cm", A_3: "Petal length in cm" and A_4="Petal width in cm". Therefore, the names of the FIs mined from each partition $\mathcal{D}_i$ are in the form of "Attribute=length/width".

Let $\alpha = 0.2$ be the minimum support threshold set by the user. The local frequent itemsets (FIs) mined in each local database partition are given in Table 13. Let us estimate the global support of each itemset $I_k \in \cup_{i=1}^{6}\{FIS(\mathcal{D}_i, 0.2)\}$ in $\mathcal{D}_{iris}$ using the formula (1) proposed in [35]. We notice that all the global itemsets synthesized from the six databases are not valid or infrequent (i.e., $\forall\, I_k \in FIS(\mathcal{D}_{iris}, 0.2), supp(I_k, \mathcal{D}) < \alpha$). Again, this is mainly due to the irrelevant data involved during the synthesizing process. Now, if we cluster the six databases using our proposed algorithm and then synthesize the global itemsets from each cluster individually, we get the following patterns: $FIS(C_1 = \{\mathcal{D}_1, \mathcal{D}_2\}, 0.2) = \{\langle A_3 = 1.5, 0.26\rangle, \langle A_3 = 1.4, 0.26\rangle, \langle A_4 = 0.2, 0.58\rangle\}$,



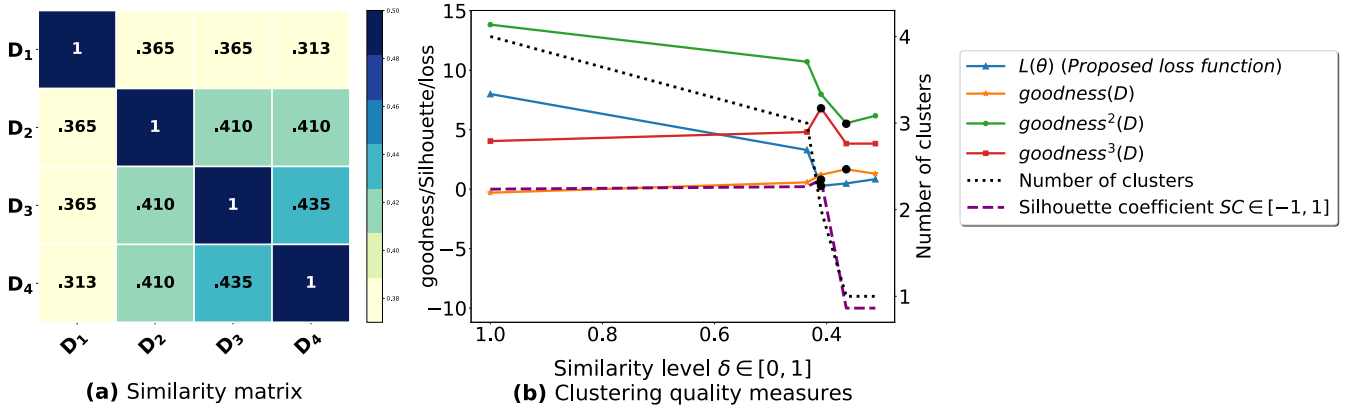
(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 6. **(a): 7×7 similarity matrix from the databases in Table 11. The mined frequent itemsets (FIs) are extracted under a minimum threshold $\alpha = 0.42$ and the pairwise similarities are obtained using (3). (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our loss function $L(\theta)$.**



(a) Similarity matrix **(b)** Clustering quality measures

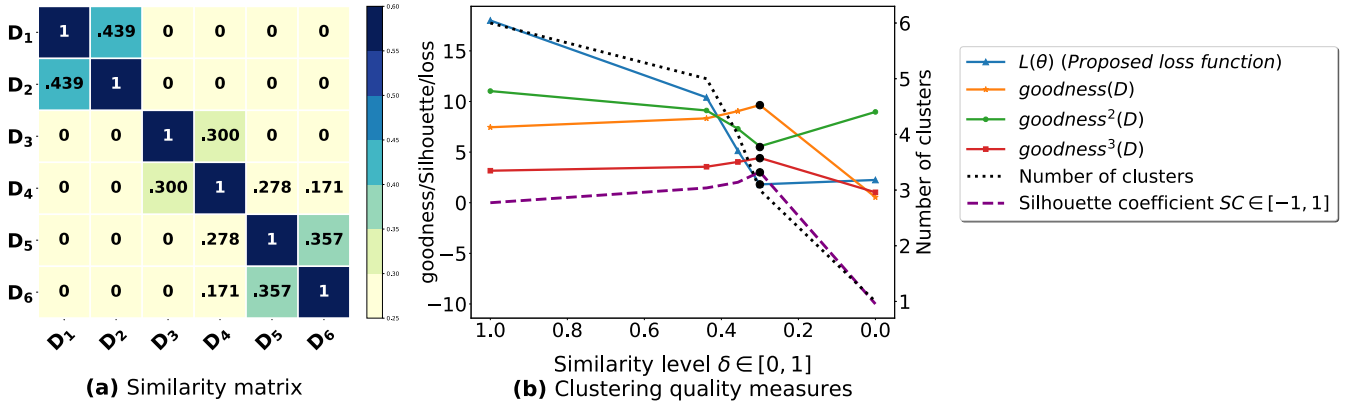
FIGURE 7. **(a): 12×12 similarity table between the twelve databases partitioned from the real dataset Zoo [68]. The mined frequent itemsets (FIs) are extracted under a minimum threshold $\alpha = 0.5$. We then apply the similarity measure (3) on the twelve FIs to get the pairwise similarities. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta) \div 10$.**



(a) Similarity matrix **(b)** Clustering quality measures

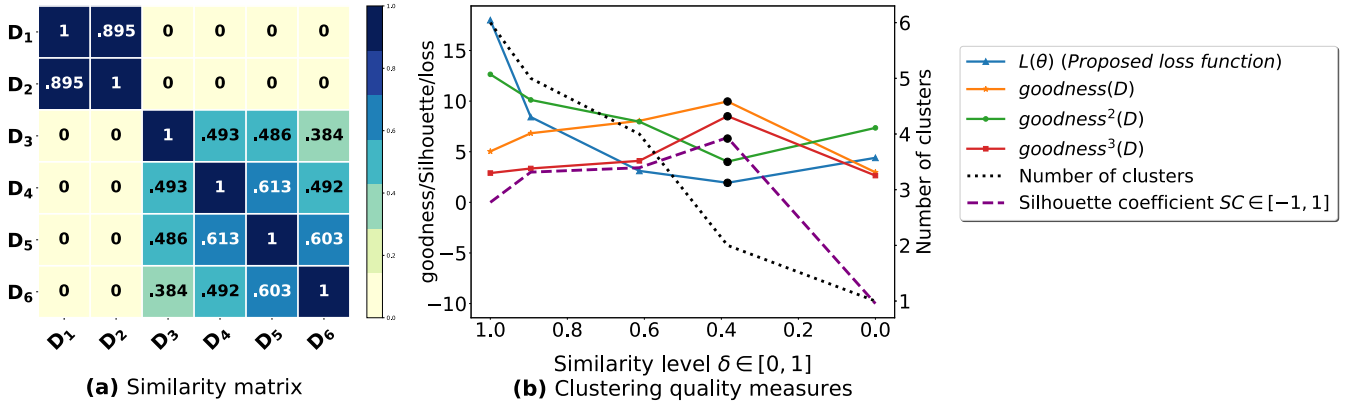
FIGURE 8. **(a): 4×4 similarity table between four databases partitioned from the real dataset Mushroom [68]. The mined frequent itemsets (FIs) are extracted under a minimum threshold $\alpha = 0.5$. We then apply the similarity measure (3) on the four FIs to get the pairwise similarities. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta)$.**

$FIS(C_2 = \{\mathcal{D}_3, \mathcal{D}_4\}, 0.2) = \{\langle A_4 = 1.3, 0.26\rangle\}$ and $FIS(C_3 = \{\mathcal{D}_5, \mathcal{D}_6\}, 0.2) = \{\langle A_2 = 3.0, 0.24\rangle, \langle A_4 = 1.8, 0.22\rangle\}$. Hence, the obtained clustering contributed in discovering novel and useful patterns from the local patterns already mined in the local databases. For instance, the new pattern $\langle A_3 = 1.5, 0.26\rangle$ in $FIS(C_1, 0.2)$ indicates that 26%



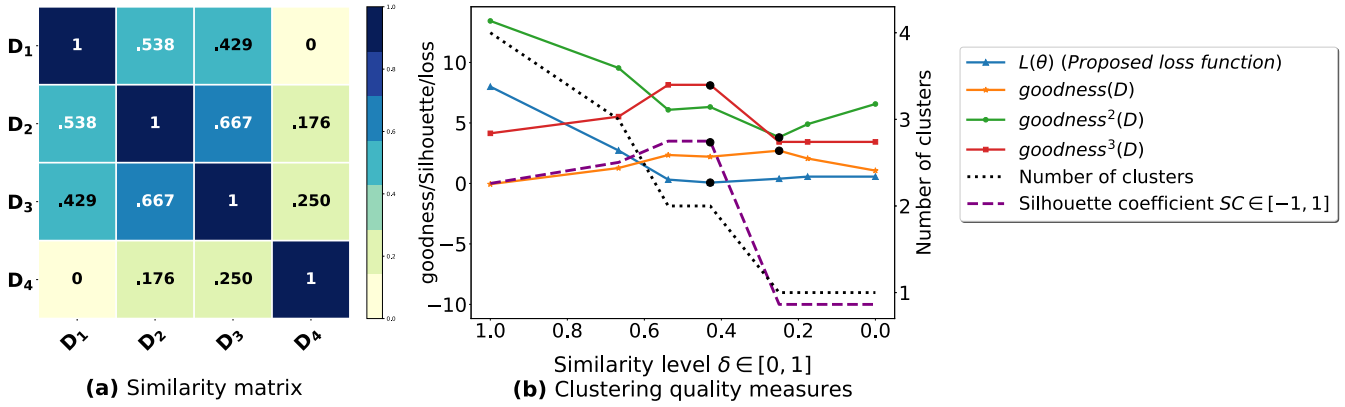
(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 9. (a): 6×6 similarity table between six databases partitioned from the real dataset Iris [68]. The mined frequent itemsets (FIs) are extracted under a minimum threshold $\alpha = 0.2$. We then apply the similarity measure (3) on the six FIs to get the pairwise similarities. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta)$.



(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 10. (a): 6×6 similarity matrix between two databases partitioned from Zoo dataset [68] and four databases partitioned from Mushroom dataset [68]. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta)$.



(a) Similarity matrix **(b)** Clustering quality measures

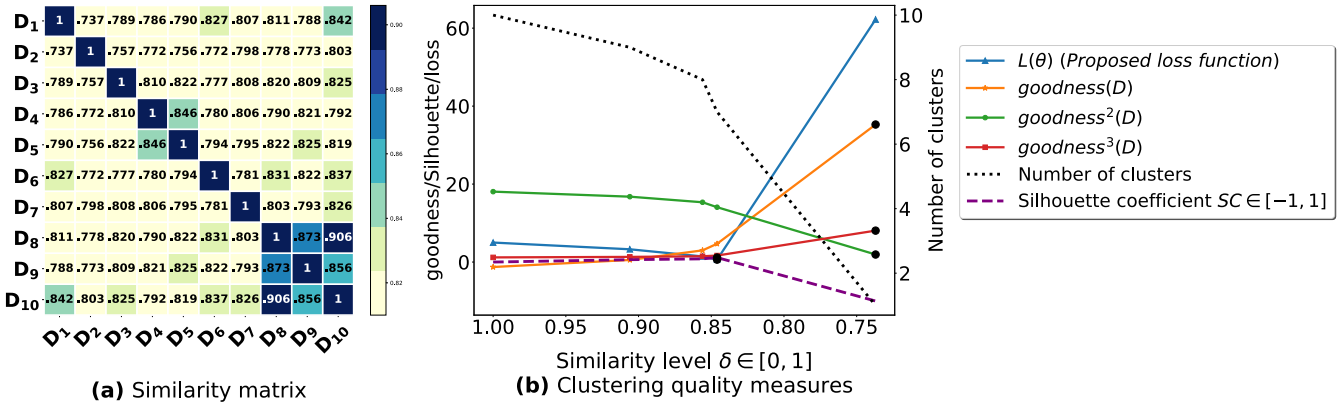
FIGURE 11. (a): 4×4 similarity matrix obtained from [36]. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta)$.

of Iris-Setosa plants have a Petal length=1.5cm. In Fig. 9 (b) and the fourth row of Table 10, all the clustering quality measures have identified the ground truth clustering at $\delta = 0.3$.

All the previous results confirm that our quasi-convex loss function has successfully identified the relevant clusters matching the ground truth and improving the valid FIs discovered from each cluster of the datasets in Table 9.

2) RUNNING TIME AND CLUSTERING PERFORMANCE

In the second part of our experiments, we are interested in comparing the clustering output and the execution time of our



(a) Similarity matrix **(b)** Clustering quality measures

FIGURE 12. **(a): 10×10 similarity table between the ten databases partitioned from the synthetic dataset T10I4D100K [64]. The mined frequent itemsets (FIs) are extracted under a minimum threshold $\alpha = 0.03$. We then apply the similarity measure (3) on the ten FIs to get the pairwise similarities. (b): represents the graphs corresponding to *goodness*($\mathcal{D}$) [22], *goodness*^{2}($\mathcal{D}$) × 10 [25], *goodness*^{3}($\mathcal{D}$) × 10 [23], the Silhouette Coefficient *SC* × 10 [39], the generated number of clusters and our proposed loss function $L(\theta) \div 10$.**

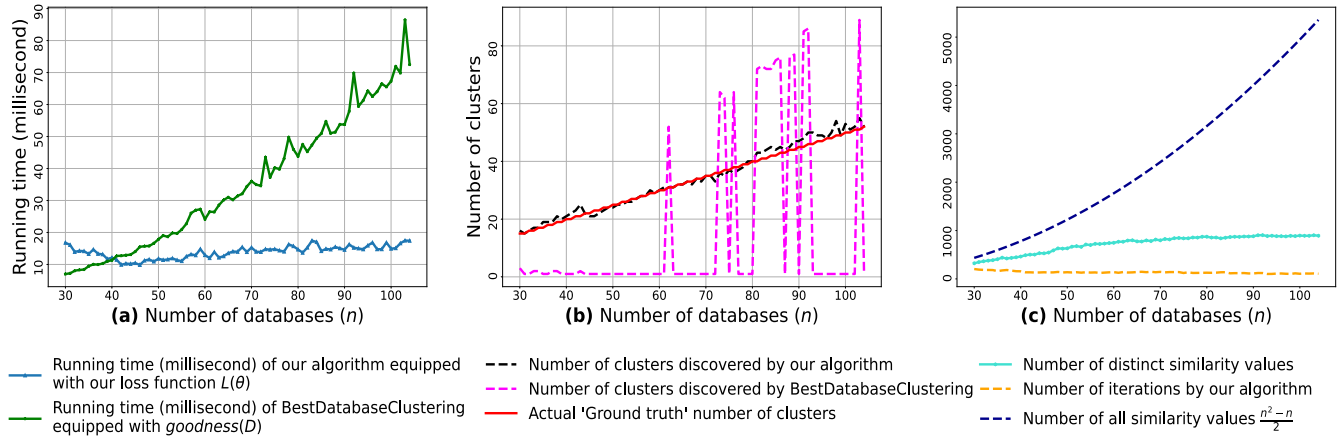


FIGURE 13. **(a): the time performance of our proposed algorithm against BestDatabaseClustering [24] on $n = 30$ to 120 random data samples under a learning rate $\eta = 0.001$. The x-axis represents the number of databases (n) in the 3 subfigures (a), (b) and (c). The y-axis in subfigure (a) represents the running time in milliseconds. (b): represents the number of clusters discovered by both algorithms and the actual number of clusters. (c): represents the number of iterations performed by our algorithm and the number of unique and total similarity values between the n current databases.**

algorithm against the clustering algorithm for multi-database mining BestDatabaseClustering [24] on the same multiple data samples. Therefore, we have used scikit-learn random sample generator (sklearn.datasets.make_blobs) [69] to generate from $n = 30$ to 120 samples, with m features ($m = 2$ by default), and with a predetermined number of centers that is equal to $\lfloor \frac{n}{2} \rfloor$. Then, each feature I_k is normalized into the range [0,1] using the *min-max scaling* method [70], which is defined as follows:

$$I_k = \frac{I_k - \min(I_k)}{\max(I_k) - \min(I_k)}, \quad for\ k = 1..m \tag{35}$$

where $\min(I_k)$ and $\max(I_k)$ are the minimum and the maximum values in the k-th column feature vector, respectively.

We then use an adapted version of *sim* (3) (i.e., instead of working on hash tables, we pass m-dimensional feature vectors as arguments to *sim*) to compute the $\binom{n}{2}$ pairwise similarity values between the n current data samples. Since the attributes I_k are not actual frequent itemsets mined from each sample, we have set the correction factor to zero. We run both algorithms on each $\binom{n}{2}$ pairwise similarity values and plot their outputs in Fig. 13. Precisely, for each $n \times n$ similarity matrix, we collect the execution time (represented in milliseconds) in Fig. 13 (a) and the number of clusters returned by both algorithms in Fig. 13 (b). Meanwhile, in Fig. 13 (c), we record the number of iterations ran by our algorithm and the number of distinct similarity values. We should note that during the experiment, we have not added up the CPU time required to compute the similarity values, since the same similarity function (3) was used for both algorithms.

As the number of data samples (n) increases, we notice a rapid increase in the execution time of BestDatabaseClustering [24]. This is due to the fact that its time complexity, estimated as $O(n^2 log_2(h))$, is strongly related to n and h (the number of distinct similarity values). Also, in order to find the optimal clustering at which *goodness*($\mathcal{D}_n$) is maximum, BestDatabaseClustering needs to browse all the $h \leq (n^2 - n)/2$ distinct similarity values due to the fact that *goodness*($\mathcal{D}_n$) is neither concave nor quasi-concave on the interval [0,1].

By studying the number of clusters returned by both algorithms in Fig. 13 (b), we observe that most of the time,

BestDatabaseClustering fails to find the relevant clusters and tend to generate either n singleton clusters or one cluster of the n data samples. This is depicted by the existence of peaks (where the curve reaches its maximum value at $f(\mathcal{D}_n) = n$) and valleys (where the curve is flat at $f(\mathcal{D}_n) = 1$) on the dashed magenta line. In fact, the clustering normalized root-mean-square error (nrmse) for BestDatabaseClustering [24] is estimated as $nrmse = 0.72$, such that:

$$nrmse = \frac{\sqrt{\frac{1}{N} \times \sum_{n=30}^{N=120} (f(\mathcal{D}_n) - f\hat{}(\mathcal{D}_n))^2}}{\lfloor \frac{120}{2} \rfloor - \lfloor \frac{30}{2} \rfloor} \tag{36}$$

where $\mathcal{D}_n$ is the generated dataset, n is the number of samples in $\mathcal{D}_n$, $f(\mathcal{D}_n)$ is the number of clusters discovered by BestDatabaseClustering [24] and $f\hat{}(\mathcal{D}_n)$ is the actual number of clusters set to $\lfloor \frac{n}{2} \rfloor$. Conversely, as n increases, our algorithm generates a number of clusters that is close to the actual number of clusters, that is, $f_\theta(\mathcal{D}_n) \approx \lfloor \frac{n}{2} \rfloor$. The clustering normalized root-mean-square error for our algorithm is $nrmse = 0.041$. Overall, the running time of our algorithm stays relatively steady with respect to n. In fact, our algorithm depends mostly on the number of iterations required to find the global minimum, which is dependent on the learning step size η and decay rate $decay\_rate$. A grid search is often conducted to sample a randomly set of points (η, $decay\_rate$) within a two-dimensional grid in order to explore the values returning the optimal weight vector θ.

Our algorithm terminates as soon as the global minimum of the loss function $L(\theta)$ has been reached. This is why the running time of our algorithm is most of the time smaller than that of BestDatabaseClustering. In some cases where the similarity values calculated for each n data samples are too small ($\simeq 0$), the corresponding weights values θ remain relatively away from the maximum value of one ($\theta_{p,q} \ll 1$). Hence, our algorithm needs some few iterations of updating operations to converge, especially if the learning rate is too small. This is the case for the $n \in [30, 40]$ data samples produced by the generator sklearn.datasets.make_blobs. To overcome the latter problem, we just need to increase the learning rate and/or reduce the decay rate when the mean of the pairwise similarity values in the current n data samples is below a certain threshold predefined by the user.

V. CONCLUSION

In this article, we presented a learning algorithm to minimize a quasi-convex loss function used as a clustering evaluation measure. Various experiments have been conducted on public, synthetic and random data samples. The proposed algorithm outperforms the existing clustering algorithm for multi-database mining and returns optimal results in terms of running time and accuracy. From the experimental results, we can see that the running of our algorithm is strongly dependent on the number of iterations needed to satisfy the optimality criterion, which is also dependent on the choice of the initial learning and decay rates. In this work, we used a standard decaying method to adjust the learning rate over time as we approach convergence. For our future work, we will explore new methods aiming to reduce the number of iterations by choosing an appropriate step size. For this purpose, we will investigate techniques such as *cyclical learning rates* for training neural networks. This allows the learning rate to cyclically oscillate between two predefined bounds, a minimum and maximum learning rates selected by the user, leading to a faster convergence. Finally, in real-world applications, clusters may overlap and database objects could belong to different clusters with different membership probabilities. Such grouping is called *soft-clustering*. As for our future work, we would be interested in proposing an extended version of the current algorithm that takes into account the fuzzy nature of cluster memberships.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments which have significantly improved the quality of this article.

REFERENCES

[1] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 962–969, Dec. 1996.
[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. Int. Conf. Very Large Data Bases*, vol. 1215, 1994, pp. 487–499.
[3] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining Knowl. Discovery*, vol. 8, no. 1, pp. 53–87, Jan. 2004.
[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.
[5] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, Sep. 1967.
[6] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 849–856.
[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[8] D. Huang, J.-H. Lai, C.-D. Wang, and P. C. Yuen, "Ensembling over-segmentations: From weak evidence to strong segmentation," *Neurocomputing*, vol. 207, pp. 416–427, Sep. 2016.
[9] Z. Wang, D. Zhang, X. Zhou, D. Yang, Z. Yu, and Z. Yu, "Discovering and profiling overlapping communities in location-based social networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 44, no. 4, pp. 499–509, Apr. 2014.
[10] C.-D. Wang, J.-H. Lai, and P. S. Yu, "NEIWalk: Community discovery in dynamic content-based networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 7, pp. 1734–1748, Jul. 2014.
[11] D. Rafailidis and P. Daras, "The TFC model: Tensor factorization and tag clustering for item recommendation in social tagging systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 43, no. 3, pp. 673–688, May 2013.
[12] P. Symeonidis, "ClustHOSVD: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 9, pp. 1240–1251, Sep. 2016.
[13] Q. Zhao, C. Wang, P. Wang, M. Zhou, and C. Jiang, "A novel method on information recommendation via hybrid similarity," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 3, pp. 448–459, Mar. 2018.
[14] M. R. Anderberg, *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*, vol. 19. New York, NY, USA: Academic, 2014.
[15] C. C. Aggarwal and C. K. Reddy, "Data clustering," *Algorithms Application*. Boca Raton, FL, USA: CRC Press, 2014.
[16] Y.-J. Zhang and Z.-Q. Liu, "Self-splitting competitive learning: A new on-line clustering paradigm," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 369–380, Mar. 2002.
[17] E. Yair, K. Zeger, and A. Gersho, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. Signal Process.*, vol. 40, no. 2, pp. 294–309, Feb. 1992.

[18] T. Hofmann and J. M. Buhmann, "Competitive learning algorithms for robust vector quantization," *IEEE Trans. Signal Process.*, vol. 46, no. 6, pp. 1665–1675, Jun. 1998.
[19] T. Kohonen, *Self-Organizing Maps*, vol. 30. Berlin, Germany: Springer, 2012.
[20] N. R. Pal, J. C. Bezdek, and E. C.-K. Tsao, "Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 549–557, Jul. 1993.
[21] J. Mao and A. K. Jain, "A self-organizing network for hyperellipsoidal clustering (HEC)," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 16–29, Jan. 1996.
[22] A. Adhikari and J. Adhikari, "Clustering multiple databases induced by local patterns," in *Advances in Knowledge Discovery in Databases*. Cham, Switzerland: Springer, 2015, pp. 305–332.
[23] Y. Liu, D. Yuan, and Y. Cuan, "Completely clustering for multi-databases mining," *J. Comput. Inf. Syst.*, vol. 9, no. 16, pp. 6595–6602, 2013.
[24] S. Miloudi, S. A. R. Hebri, and S. Khiat, "Contribution to improve database classification algorithms for multi-database mining," *J. Inf. Process. Syst.*, vol. 14, no. 3, pp. 709–726, 2018.
[25] H. Tang and Z. Mei, "A simple methodology for database clustering," in *Proc. 5th Int. Conf. Comput. Eng. Netw.*, Oct. 2015, p. 019.
[26] R. Wang, W. Ji, M. Liu, X. Wang, J. Weng, S. Deng, S. Gao, and C.-A. Yuan, "Review on mining data from multiple data sources," *Pattern Recognit. Lett.*, vol. 109, pp. 120–128, Jul. 2018.
[27] C. Lemaréchal, "Cauchy and the gradient method," *Doc Math Extra*, vol. 251, p. 254, Dec. 2012.
[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[29] S. Zhang and M. J. Zaki, "Mining multiple data sources: Local pattern analysis," *Data Mining Knowl. Discovery*, vol. 12, nos. 2–3, pp. 121–125, May 2006.
[30] A. Adhikari and P. R. Rao, "Synthesizing heavy association rules from different real data sources," *Pattern Recognit. Lett.*, vol. 29, no. 1, pp. 59–71, Jan. 2008.
[31] A. Adhikari and J. Adhikari, *Advances in Knowledge Discovery in Databases*. Cham, Switzerland: Springer, 2015.
[32] A. Adhikari, L. C. Jain, and B. Prasad, "A state-of-the-art review of knowledge discovery in multiple databases," *J. Intell. Syst.*, vol. 26, no. 1, pp. 23–34, Jan. 2017.
[33] S. Zhang, C. Zhang, and X. Wu, "Identifying high-vote patterns," *Knowl. discovery multiple databases*, pp. 157–183, 2004.
[34] S. Zhang, C. Zhang, and X. Wu, "Identifying exceptional patterns," in *Proc. Knowl. Discovery Multiple Databases*, 2004, pp. 185–195.
[35] T. Ramkumar and R. Srinivasan, "Modified algorithms for synthesizing high-frequency rules from different data sources," *Knowl. Inf. Syst.*, vol. 17, no. 3, pp. 313–334, Dec. 2008.
[36] X. Wu, C. Zhang, and S. Zhang, "Database classification for multi-database mining," *Inf. Syst.*, vol. 30, no. 1, pp. 71–88, Mar. 2005.
[37] H. Li and Z. Hang, "An improved database classification algorithm for multi-database mining," in *Frontiers Algorithmics* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2009, pp. 346–357.
[38] S. Na, L. Xumin, and G. Yong, "Research on K-means clustering algorithm: An improved k-means clustering algorithm," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Informat.*, Apr. 2010, pp. 63–67.
[39] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.
[40] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A new data clustering algorithm and its applications," *Data Mining Knowl. Discovery*, vol. 1, no. 2, pp. 141–182, 1997.
[41] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Trans. Pattern Anal. Mach. Intell.*, vols. PAMI–6, no. 1, pp. 81–87, Jan. 1984.
[42] D. Huang, J.-H. Lai, and C.-D. Wang, "Robust ensemble clustering using probability trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1312–1326, May 2016.
[43] D. Huang, C.-D. Wang, and J.-H. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1460–1473, May 2018.
[44] D. Huang, C.-D. Wang, H. Peng, J. Lai, and C.-K. Kwoh, "Enhanced ensemble clustering via fast propagation of cluster-wise similarities," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Nov. 6, 2020, doi: 10.1109/TSMC.2018.2876202.
[45] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1212–1226, Jun. 2020.
[46] Y. Énbabaoálu, G. Michailidis, and J. Z. Li, "Critical limitations of consensus clustering in class discovery," *Sci. Rep.*, vol. 4, no. 1, May 2015, Art. no. 6207.
[47] Y. Djenouri, J. Chun-Wei Lin, K. Norvag, and H. Ramampiaro, "Highly efficient pattern mining based on transaction decomposition," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1646–1649.
[48] A. Savasere, E. R. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GIT-CC-95-04, 1995.
[49] S. Zhang and X. Wu, "Large scale data mining based on data partitioning," *Appl. Artif. Intell.*, vol. 15, no. 2, pp. 129–139, Feb. 2001.
[50] C. Zhang, M. Liu, W. Nie, and S. Zhang, "Identifying global exceptional patterns in multi-database mining," *IEEE Intell. Inform. Bull.*, vol. 3, no. 1, pp. 19–24, Dec. 2004.
[51] S. Zhang, C. Zhang, and J. X. Yu, "An efficient strategy for mining exceptions in multi-databases," *Inf. Sci.*, vol. 165, nos. 1–2, pp. 1–20, Sep. 2004.
[52] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 372–390, 2000.
[53] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *Proc. FIMI*, vol. 126, 2004, pp. 1–11.
[54] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, "H-mine: Hyper-structure mining of frequent patterns in large databases," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2001, pp. 441–448.
[55] S. Zhang, X. Wu, and C. Zhang, "Multi-database mining," *IEEE Comput. Intell. Bull.*, vol. 2, no. 1, pp. 5–13, Jun. 2003.
[56] C. Zhang, J. X. Yu, and S. Zhang, "Identifying interesting patterns in multidatabases," in *Classification and Clustering for Knowledge Discovery*. Berlin, Germany: Springer, 2005, pp. 91–112.
[57] X. Wu and S. Zhang, "Synthesizing high-frequency rules from different data sources," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 2, pp. 353–367, Mar. 2003.
[58] S. Zhang, X. You, Z. Jin, and X. Wu, "Mining globally interesting patterns from multiple databases using kernel estimation," *Expert Syst. Appl.*, vol. 36, no. 8, pp. 10863–10869, Oct. 2009.
[59] U. Yun, "Efficient mining of weighted interesting patterns with a strong weight and/or support affinity," *Inf. Sci.*, vol. 177, no. 17, pp. 3477–3499, Sep. 2007.
[60] R. Srinivasan and T. Ramkumar, "The effect of correction factor in synthesizing global rules in a multi-database mining scenario," *J. Appl. Comput. Sci. Math.*, vol. 3, no. 6, pp. 33–38, 2009.
[61] D. Pavlov, H. Mannila, and P. Smyth, "Probabilistic models for Query approximation with large sparse binary data sets," in *Proc. 16th Conf. Uncertainty Artif. Intell.* Burlington, MA, USA: Morgan Kaufmann, 2000, pp. 465–472.
[62] K. Salim, B. Hafida, and R. S. Ahmed, "Probabilistic models for local patterns analysis," *J. Inf. Process. Syst.*, vol. 10, no. 1, pp. 145–161, Mar. 2014.
[63] L. Kaufman and P. J. Rousseeuw, *Finding Groups Data: Introduction to Cluster Analysis*, vol. 344. Hoboken, NJ, USA: Wiley, 2009.
[64] IBM Almaden Quest research group. *Frequent Itemset Mining Dataset Repository*. Accessed: Oct. 10, 2019. [Online]. Available: http://fimi.ua.ac.be/data/
[65] J. C. Platt and A. H. Barr, "Constrained differential optimization," in *Proc. Neural Inf. Process. Syst.*, 1988, pp. 612–621.
[66] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[67] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Data structures for disjoint sets," in *Proc. Introduction Algorithms*, 2009, pp. 498–524.
[68] Center for Machine Learning and Intelligent Systems. *UCI Machine Learning Repository*. Accessed: Oct. 10, 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/
[69] G. Thirion, G. Varoquaux, A. Gramfort, V. Michel, O. Grisel, G. Louppe, and J. Nothman, *Scikit-Learn: Sklearn Datasets Makeblobs*. Accessed: Oct. 10, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html
[70] A. Gramfort, M. Blondel, O. Grisel, A. Mueller, E. Martin, G. Patrini, and E. Chang. *Scikit-Learn: Sklearn Preprocessing Minmaxscaler*. Accessed: Oct. 10, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

SALIM MILOUDI (Member, IEEE) was born in Algeria in 1987. He received the B.S. and M.S. degrees in information system engineering from the University of Science and Technology-Mohamed Boudiaf (USTO-MB), Oran, Algeria, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree in computer science with Wuhan University, China.

From 2011 to 2012, he was a Teaching Assistant with USTO-MB. From 2014 to 2016, he was an Information System Engineer with the Health Care Facility, Oran. His research interests include data mining for knowledge discovery, multi-database mining, and machine learning.

YULIN WANG (Senior Member, IEEE) was born in China in 1966. He received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 1987, the M.S. degree in electronic and information engineering from the Huazhong University of Science and Technology, Wuhan, in 1997, and the Ph.D. degree in multimedia information processing from the University of London, U.K., in 2005.

He was a Research Fellow with the Wuhan Ship Communication Institute, China, for six years, and a Senior Engineer Huawei Technology Company, Ltd., China, for three years. He is currently a Full Professor with the School of Computer Science, Wuhan University, China. He was an Expert of the National High Technology Research and Development Program of China. He has authored or coauthored three books, 30 conference papers, and 55 journal articles. His research interests include digital rights management, digital watermarking, multimedia and network security, and signal processing. He was the Editor-in-Chief of the journal *Advances in Multimedia* and the journal *Image and Graphics*. He was invited as a keynote speaker in more than 16 international conferences.

Prof. Wang is the Deputy Director of the Hubei Science and Technology Committee, China, and a Senior Member of China Computer Federation.

WENJIA DING (Member, IEEE) was born in China in 1994. She received the B.S. degree (Hons.) and the M.S. degree in computer science from the National University of Singapore, Singapore, in 2015 and 2017, respectively. She is currently pursuing the Ph.D. degree with Wuhan University, China.

She was a Junior Researcher with IBM Singapore Pte., Ltd., during her internship in 2015. She jointly authored or coauthored five academic papers and held two patents. Her main research interests include image and video processing, information and network security, and computer vision.

Miss Ding is a member of the International Academy of Computer Technology and the South Asia Institute of Science and Engineering.