An Improved Similarity-Based Clustering Algorithm for Multi-Database Mining

Salim Miloudi *, Yulin Wang * and Wenjia Ding *

School of Computer Science, Wuhan University, Wuhan 430072, China; salim.miloudi@whu.edu.cn (S.M.); yulinwang@whu.edu.cn (Y.W.); w.j.ding@whu.edu.cn (W.D.)

Abstract: Clustering algorithms for multi-database mining (MDM) rely on computing \(\binom{n^2 - n}{2}\) pairwise similarities between \(n\) multiple databases to generate and evaluate \(m \in [1, \binom{n^2 - n}{2}]\) candidate clusterings in order to select the ideal partitioning that optimizes a predefined goodness measure. However, when these pairwise similarities are distributed around the mean value, the clustering algorithm becomes indecisive when choosing what database pairs are considered eligible to be grouped together. Consequently, a trivial result is produced by putting all the \(n\) databases in one cluster or by returning \(n\) singleton clusters. To tackle the latter problem, we propose a learning algorithm to reduce the fuzziness of the similarity matrix by minimizing a weighted binary entropy loss function via gradient descent and back-propagation. As a result, the learned model will improve the certainty of the clustering algorithm by correctly identifying the optimal database clusters. Additionally, in contrast to gradient-based clustering algorithms, which are sensitive to the choice of the learning rate and require more iterations to converge, we propose a learning-rate-free algorithm to assess the candidate clusterings generated on the fly in fewer upper-bounded iterations. To achieve our goal, we use coordinate descent (CD) and back-propagation to search for the optimal clustering of the \(n\) multiple database in a way that minimizes a convex clustering quality measure \(\mathcal{L}(\theta)\) in less than \((n^2 - n)/2\) iterations. By using a max-heap data structure within our CD algorithm, we optimally choose the largest weight variable \(\theta_{i,q}\) at each iteration \(i\) such that taking the partial derivative of \(\mathcal{L}(\theta)\) with respect to \(\theta_{i,q}\) allows us to attain the next steepest descent minimizing \(\mathcal{L}(\theta)\) without using a learning rate. Through a series of experiments on multiple database samples, we show that our algorithm outperforms the existing clustering algorithms for MDM.

Keywords: coordinate descent; clustering; multi-database mining; fuzziness; binary entropy loss; similarity matrix

1. Introduction

Large multi-branch companies need to analyze multiple databases to discover useful patterns for the decision-making process. To make global decisions for the entire company, the traditional approach suggests to merge and integrate the local branch-databases into a huge data warehouse, and then one can apply data mining algorithms [1] to the accumulated dataset to mine the global patterns useful for all the branches of the company. However, there are some limitations associated with this approach. For instance, the cost of moving the data over the network, and integrating and storing potentially heterogeneous databases could be expensive. Moreover, some branches may not accept sharing their raw data due to the underlying privacy issues. More crucially, integrating a large amount of irrelevant data can easily disguise some essential patterns hidden in multiple databases. To tackle the latter problems, it is suggested to keep the transactional data stored locally and only forward the local patterns mined at each branch database to a central site where they will be clustered into disjoint cohesive pattern-base groups for knowledge discovery. In fact, analyzing the local patterns present in each individual cluster of the multiple databases (MDB) enhances the quality of aggregating novel relevant patterns, and also facilitates...
the parallel maintenance of the obtained database clusters. Various clustering algorithms and models have been introduced in the literature, namely spectral-based models [2], hierarchical [3], partitioning [4], competitive learning-based models [5–7] and artificial neural networks (ANNs) based clustering [8–10]. Additionally, clustering could be applied in many domains [11,12] including community discovery in social networks [13,14], image segmentation [15,16] and recommendation systems [17–19]. In this article, we focus on exploring similarity-based clustering models for multi-database mining [20–23], due to their stability, simplicity [24] and robustness in partitioning graphs of $n$ multiple databases into $k$ connected components consisting of similar database objects. Nevertheless, the existing clustering quality measures in [20–23] are non-convex objectives suffering from the existence of local optima. Consequently, identifying the optimal clustering may be a difficult task, as it requires evaluating all the candidate clusterings generated at all the local optima in order to find the ideal clustering.

To address the issues associated with clustroid initialization, preselection of a suitable number of clusters and non-convexity of the clustering quality objectives, we proposed in [25,26] an algorithm named GDMDBCWith, which minimizes a quasi-convex loss function quantifying the quality of the multi-database clustering, without a priori assumptions about which number of clusters should be chosen. Therefore, in contrast to the clustering models proposed in [20–23], GDMDBCWith [25] does not require us to produce and assess all the possible candidate classifications in order to find the optimal partitioning. Alternatively, each partitioning is assessed on the fly as it is generated and the clustering algorithm terminates right after attaining the global minimum of the objective function. However, the existing gradient-based clustering algorithms [25,26] are strongly dependent on the choice of the learning rate $\eta$, which influences the number of learning cycles required to find the optimal partitioning. In fact, selecting a larger $\eta$ value may cause global minimum overshooting and setting a smaller $\eta$ value may necessitate many learning iterations for the algorithm to converge.

In this paper, we improve upon previous work [25,26] and propose a learning-rate-free (i.e., independent of the learning rate $\eta$) algorithm requiring fewer upper-bounded iterations (i.e., the maximum number of iterations is at most $(n^2 - n)/2$) to minimize a clustering convex loss function $\mathcal{L}(\theta)$ using coordinate descent (CD) and back-propagation. Precisely, our proposed algorithm minimizes a quadratic hinge-based loss $\mathcal{L}(\theta)$ over the first largest coordinate variable $\theta_{p,q}$ while keeping the rest of the $(\binom{n}{2} - 1)$ variables fixed. Then, it minimizes $\mathcal{L}(\theta)$ over the second largest coordinate variable while keeping the rest of the $(\binom{n}{2} - 1)$ variables fixed, and so on until convergence or until cycling through all the $(\binom{n}{2})$ coordinate variables. Consequently, our algorithm becomes faster than GDMDBCWith [25] which is dependent on a learning rate and also requires us to minimize the cost over a large set of variables at each iteration. This can be a very challenging problem in contrast to minimizing the loss over a single variable at a time while keeping all the other dimensions fixed.

On the other hand, existing clustering algorithms for multi-database mining (MDM) [20–23,25,26] proceed by computing $(n^2 - n)/2$ pairwise similarities $\text{sim}(D_p, D_q) \in [0,1]$ between $n$ multiple databases, and then use these values to generate and evaluate $m \in [1, (n^2 - n)/2]$ candidate clusterings in order to select the ideal partitioning optimizing a given goodness measure. However, when $\text{sim}(D_p, D_q)_{n \times n}$ ($p = 0, \ldots, n-2; q = p+1, \ldots, n-1$) are distributed around the mean value $\mu = 0.5$, the fuzziness index of the similarity matrix increases and the clustering algorithm becomes uncertain when choosing what database pairs are considered similar and hence eligible to be put into the same cluster. Consequently, a trivial result is produced, i.e., putting all the $n$ databases in one cluster or returning $n$ singleton clusters. To tackle the latter problem, we propose a learning algorithm to reduce the fuzziness in the pairwise similarities by minimizing a weighted binary entropy loss function $\mathcal{H}(\cdot)$ via gradient descent and back-propagation. Precisely, the learned model will force the similarity values above 0.5 to go closer to their maximum value ($\approx 1$), and let those below 0.5 go closer to their minimum value ($\approx 0$) in a way that minimizes $\mathcal{H}(\cdot)$. This will significantly reduce the associated fuzzi-
ness and improve the certainty of the clustering algorithm to correctly identify the optimal database clusters. The main contributions of this article are listed as follows:

- Unlike the existing algorithms proposed in [20–23,25,26] where one-class trivial clusterings are produced when the similarity values are centered around the mean value, we have added a preprocessing layer prior to clustering where the pairwise similarities are adjusted to reduce the associated fuzziness and hence improve the quality of the produced clustering. Our experimental results show that reducing the fuzziness of the similarity matrix helps generating meaningful relevant clusters that are different to the one-class trivial clusterings.

- Unlike the multi-database clustering algorithms proposed in [20–23], our approach uses a convex objective function $L(\theta)$ to assess the quality of the produced clustering. This allows our algorithm to terminate just after attaining the global minimum of the objective function (i.e., after exploring fewer similarity levels). Consequently, this avoids generating unnecessary candidate clusterings, and hence reduces the CPU overhead. On the other hand, the clustering algorithms in [20–23] use non-convex objectives (i.e., they suffer from the existence of local optima due to the use of more than two monotonic functions), and therefore require generating and evaluating all the $(n^2 - n)/2$ local candidate clustering solutions in order to find the clustering located at the global optimum.

- Furthermore, unlike the previous gradient-based clustering algorithms [25,26], our proposed algorithm is leaning-rate-free (i.e., independent of the learning rate), and needs at most (in the worst case) $(n^2 - n)/2$ iterations to converge. That is why our proposed algorithm is faster than GDMDBClustering [25], which is strongly dependent on the learning step size $\eta$ and its decay rate.

- Additionally, unlike the similarity measure proposed in [20], which assumes that the same threshold was used to mine the local patterns from the $n$ transactional databases, our proposed similarity measure takes into account the existence of $n$ different local thresholds, which are then combined to calculate a new threshold for each cluster. Afterward, using the new thresholds, our similarity measure accurately estimates the valid patterns post-mined from each cluster in order to compute the $(n^2 - n)/2$ pairwise similarities.

- The experiments carried out on real, synthetic and randomly generated datasets show that the proposed clustering algorithm outperforms the compared clustering models in [20–23,25,26], as it has the shorted average running time and the lowest average clustering error.

The remainder of this paper is organized as follows: Section 2 presents an example motivating the importance of clustering for multi-database mining (MDM) and also reviews traditional clustering algorithms for MDM. Section 3 defines the main concepts related to similarity-based clustering and then introduces the proposed approach and its main components. Section 4 presents and analyzes the experimental results. Finally, Section 5 draws conclusions and highlights potential future work.

2. Motivation and Related Work

2.1. Motivating Example

Prior to mining the multiple databases (MDB) of a multi-branch enterprise, it is essential to cluster these MDB into disjoint and cohesive pattern-base groups sharing an important number of local patterns in common. Then, using local pattern analysis and pattern synthesizing techniques [27–30], one can examine the local patterns in each individual cluster to discover novel patterns, including the exceptional patterns [31] and the high-vote patterns [32], which are extremely useful when it comes to making special targeted decisions regarding each cluster of branches of the same corporation. In the following example, we show the impact of clustering the multi-databases of a multi-branch corporation prior to multi-database mining. Consider the six transactional databases $D = \cup_{p=1}^{6} \{D_p\}$ shown in Table 1, where each database $D_p$ records a set of transactions enclosed in parentheses and
each transaction contains a set of items separated by commas. Consider a minimum support
threshold $\alpha = 0.5$. The local frequent itemsets, denoted by $FIS(D_p, \alpha)$, and discovered from
each database $D_p$ are shown in Table 2, such that $I_k$ in each tuple $(I_k, supp(I_k, D_p))$ of $FIS(D_p, \alpha)$ is the frequent itemset name and $supp(I_k, D_p)$, named support, is the ratio of the
number of transactions in $D_p$ containing $I_k$ to the total number of transactions in $D_p$. 

Table 1. Six transactional databases $D_p$, for $p = 1, \ldots, 6$.

| Transactional Database ($D_p$) | Transactions/Rows |
|-------------------------------|-------------------|
| $D_1$                         | $(A, C), (A, B, C), (B, C), (A, B, C, D)$ |
| $D_2$                         | $(A, B, C), (B, C), (A, B), (A, C), (A, B, D)$ |
| $D_3$                         | $(B, C), (A, D), (B, C, D), (A, B, C)$ |
| $D_4$                         | $(E, F, H), (F, H), (F, G, H, I, J)$ |
| $D_5$                         | $(E, J), (F, H, J), (E, F, H, J), (F, H)$ |
| $D_6$                         | $(E, I), (E, F, H), (F, H, I, J), (E, H, J)$ |

Table 2. The frequent itemsets (FIs) discovered from each transactional database in Table 1 under a threshold $\alpha = 0.5$.

| Transactional Database ($D_p$) | Frequent Itemsets: $FIS(D_p, \alpha)$ |
|-------------------------------|----------------------------------|
| $D_1$                         | $(AC, 0.75), (AB, 0.5), (ABC, 0.5), (BC, 0.75), (C, 1.0), (B, 0.75), (A, 0.75)$ |
| $D_2$                         | $(AB, 0.6), (B, B, 0.8), (A, 0.8)$ |
| $D_3$                         | $(BC, 0.75), (D, 0.5), (C, 0.75), (B, 0.75), (A, 0.5)$ |
| $D_4$                         | $(H, 1.0), (F, 1.0), (FH, 1.0)$ |
| $D_5$                         | $(E, 0.5), (E, 0.5), (J, 0.75), (H, 0.5), (FH, 0.5), (F, 0.5), (H, 0.75), (FH, 0.75), (F, 0.75)$ |
| $D_6$                         | $(I, 0.5), (J, 0.5), (H, 0.5), (F, 0.5), (FH, 0.5), (E, 0.75), (EH, 0.5), (H, 0.75)$ |

Now, the global support of each itemset $I_k \in \cup_{p=1}^{6} \{FIS(D_p, 0.5)\}$ is calculated via the synthesizing equation [33] defined as follows:

$$supp(I_k, D) = \frac{\sum_{p=1}^{n} |D_p| \times supp(I_k, D_p)}{\sum_{p=1}^{n} |D_p|}$$

(1)

where $n = 6$ is the total number of databases in $D$ and $|D_p|$ is the number of transactions in $D_p$. For instance, we can calculate the global support of the itemset $A$ as follows:

$$supp(A, D) = \frac{0.75 \times 4 + 0.8 \times 5 + 0.5 \times 4 + 0 \times 3 + 0 \times 4 + 0 \times 4}{4 + 5 + 4 + 3 + 4 + 4}$$

$$= 0.375 < \alpha$$

After computing the global supports of the rest of the itemsets using (1), no single novel pattern has been found, i.e., $\forall I_k \in \cup_{p=1}^{6} \{FIS(D_p, 0.5)\}, supp(I_k, D) < 0.5$. The reason is that irrelevant patterns were involved in the synthesizing procedure. Now, if we examine the frequent itemsets in Table 2, we observe that some databases share many patterns in common. Precisely, the six databases seem to form two clusters, $C_1 = \{D_1, D_2, D_3\}$ and $C_2 = \{D_4, D_5, D_6\}$, where each cluster of databases tend to share similar frequent itemsets.

Next, let us use the synthesizing equation (1) on the frequent itemsets coming from every single cluster $C_i$, such that $4 \leq p \leq 6 = n$ for cluster $C_2$ and $1 \leq p \leq 3 = n$ for cluster $C_1$. This time, new valid frequent itemsets having a support value above the minimum threshold $\alpha$ are discovered in the two clusters. In fact, $FIS(C_2, 0.5) = \{(F, H, 0.727), (F, 0.727), (H, 0.818)\}$ and $FIS(C_1, 0.5) = \{(C, 0.769), (B, 0.769), (A, 0.692)\}$. The obtained patterns show that a percentage of more than 69% of the total transactions in the cluster $C_1$ include the itemsets $C,$
More than 72% of the total transactions in the cluster $C_2$ include $F \cdot H$, $F$ and $H$. Moreover, some associations between itemsets could be derived as well, such that the itemset $(F, H, 0.727) \in \text{FIS}(C_2, 0.5)$ suggests that on average, if a customer collects the item $H$ at one of the branches in $C_2$, they are likely to also buy the item $F$ with a support $\text{supp}(FH, C_2) = \text{supp}(H, C_2) = 88.87\%$ confidence.

The above example demonstrates the importance of clustering the multi-databases into disjoint cohesive clusters before synthesizing the global patterns. In fact, when the local patterns mined from the six databases were analyzed all together, no global pattern could be synthesized. On the other hand, when the six databases were divided into two different clusters and then each cluster was analyzed individually, useful and novel patterns (knowledge) were discovered. Actually, from the discovered knowledge, decision makers and stakeholders are going to have a clear vision about the branches that exhibit similar purchasing behaviors, and hence take useful decision accordingly. In fact, appropriate business decisions may be taken regarding each group of similar branches in order to predict potential purchasing patterns, increase the customer retention rate and convince customers to purchase more services in the future. Consequently, exploring and examining individual clusters of similar local patterns is going to help the discovery of new and relevant patterns capable of improving the decision-making quality.

### 2.2. Prior Work

The authors in [34] have adopted a divide and conquer mono-database mining approach to accelerate mining global frequent itemsets (FIs) in large transactional databases. In [35, 36], the authors have proposed similar work where big transactional databases are divided into $k$ disjoint transaction partitions whose sizes are small enough to be read and loaded to the random access memory. Then, the frequent itemsets (FIs) mined from all the $k$ partitions are synthesized into global FIs using an aggregation function such as the one suggested by the authors in [33]. It is worth noting that for mono-database mining applications, we usually have direct access to the raw data stored in big transactional databases. On the other hand, for multi-database mining (MDM) applications, it is suggested to keep the transactional data stored locally and only forward the local patterns mined at each branch database to a central site where they will be clustered into disjoint cohesive pattern-base groups for knowledge discovery. As a result, the confidential raw data are kept safe, and also the cost associated with transmitting a large amount of data over the network is cut off. Hence, in contrast to clustering the transactional data stored in a single data warehouse, our approach consists of clustering the local patterns mined and forwarded from multi-databases without requiring the number of clusters to be set a priori. Our purpose is to identify the group of databases that share similar patterns, such as the high-vote patterns [32] and the exceptional patterns [31,37,38] that can be used to make specific decisions regarding their corresponding branches. In the traditional clustering approach [34–36] applied for mono-database mining, we can only mine the global patterns that are supported by the whole multi-branch company.

The existing clustering algorithms for multi-database [20,21,23,39,40] are based on an agglomerative process that generates hierarchical partitionings at different levels of similarity, where each cluster in a given candidate partitioning is included in another cluster of a partitioning produced at the next similarity level. Regardless of the latter observation, each candidate partitioning is produced without taking into account the use of the clusters generated at the previous similarity levels. As a result, the clustering algorithms in [20,21,23,39,40] unnecessarily reconstruct clusters that have been built at the previous similarity levels. The latter limitation inspired the authors in [22] to design a graph-based algorithm, which maintains the classes produced at prior similarity levels in order to produce new subsequent classes out of them. Despite the fact that the experiments done in [22] showed promising results against the prior work [20,21,23,39,40], these algorithms are based on non-convex functions to evaluate the quality of the produced candidate
clusterings. Consequently, finding the ideal clustering for which a non-convex function is optimal may be a difficult problem to solve in a short time.

To face the latter problem, the authors in [26] have transformed the clustering problem into a quasi-convex optimization problem solvable via gradient descent and back-propagation. Consequently, an early stopping of the clustering process occurs right after converging to the global minimum. Hence, by avoiding generating and evaluating unnecessary candidate clusterings, we can significantly reduce the CPU execution time. Even though traditional clustering algorithms such as k-means [4,41] are intuitive, popular and not hard to implement, they remain sensitive to clustroid initialization, preselection of a suitable number of clusters and non-convexity of the clustering quality objective [42]. The silhouette plot [43] could be used to find an appropriate number of clusters, but this requires executing k-means multiple times with different numbers of clusters in order to find the ideal partitioning maximizing the silhouette objective. As a result, the time performance will be influenced in the case of clustering big high-dimensional datasets. Slightly different, hierarchical-based clustering algorithms [3] build nested hierarchical levels to visualize the relationships between different objects in the form of dendrograms. Then, it is up to the domain expert or to some non-convex metrics to determine at which level the tree diagram should be cut.

Conversely, the optimization problem formulated in [25,26] is quasi-convex. Therefore, convergence to the global optimum is independent of the initial settings. Furthermore, the proposed gradient-based clustering GDMDBC lustering [25] does not need to have the number of clusters as a parameter. Alternatively, the number of clusters becomes a parametric function in the main objective. However, GDMDBC lustering is based on the choice of a suitable learning rate, i.e., choosing a small learning rate \( \eta \) may increase the number of iterations and slow down learning the optimal weights, whereas a large \( \eta \) may let the algorithm overshoot the global minimum. To overcome the latter limitation, we propose in this paper a learning-rate-free clustering algorithm, named CDClustering, which minimizes a convex objective function quantifying the clustering quality. For this purpose, we use coordinate descent (CD) and back-propagation to search for the optimal clustering of \( n \) multiple database in less than \( (n^2 - n)/2 \) iterations and without using a learning rate. This makes our algorithm faster than the previous gradient-based clustering algorithm [25,26] which remains dependent on a learning rate defined based on some prior knowledge of the properties of the loss function. On the other hand, due to the fuzziness of the similarity matrix, which increases when the pairwise similarities are distributed around the mean value, the clustering algorithm becomes indecisive when grouping similar databases together. To face this problem, we design a learning algorithm to adjust the pairwise similarities between \( n \) multiple databases, in a way which minimizes a binary entropy loss function quantifying the fuzziness associated with the similarity matrix. Thus, the proposed algorithm becomes crisp in discriminating between the different database clusters.

3. Materials and Methods

In this section, we present our fuzziness reduction model applied to the pairwise similarities between \( n \) multiple databases and describe in details our coordinate descent-based clustering approach. Some definitions and notions relevant to this work need to be presented first.

3.1. Background and Relevant Concepts

In this subsection, we define the similarity measure between two transaction databases and present the process of generating and evaluating a given candidate clustering. We also define four clustering validity functions used to evaluate the clustering quality.
3.1.1. Similarity Measure

Each transactional database \( D_p \) is encoded as a hash-table to be defined as follows:

\[
FIS(D_p, \alpha_p) = \{ \cup_{k=1}^{m} \{ I_k, supp(I_k, D_p) \} \mid supp(I_k, D_p) \geq \alpha_p \}
\]  

where \( p = 0, \ldots, n - 1 \), \( n \) is the number of transactional databases, \( m \) is the number of frequent itemsets in \( D_p \), \( I_k \) is the name of the \( k \)-th frequent itemset, \( supp(I_k, D_p) \in [0, 1] \) is the support of \( I_k \), which is the ratio of the number of rows in \( D_p \) containing \( I_k \) to the total number of rows in \( D_p \), and \( \alpha_p \in [0, 1] \) is the minimum support threshold corresponding to \( D_p \), such that \( supp(I_k, D_p) \geq \alpha_p \). In this paper, FP-Growth [1] algorithm is used to mine the frequent itemsets in each database \( D_p \) as it only requires two passes over the whole database. Our proposed similarity measure is based on maximizing the number of global frequent itemsets (FIs) synthesized from the local FIs in each cluster. Precisely, to measure the similarity between two transactional databases \( D_q \) and \( D_p \), for \( p = 0, \ldots, n - 2, q = p + 1, \ldots, n - 1 \), we define the following function:

\[
sim(D_p, D_q) = \frac{\sum_{k,l \in \{FIS(D_p, \alpha_p) \cap FIS(D_q, \alpha_q)\}} \Psi(I_k, \{D_p, D_q\})}{|FIS(D_p, \alpha_p) \cup FIS(D_q, \alpha_q)|}
\]

where

\[
\Psi(I_k, \{D_p, D_q\}) = \begin{cases} 
1, & \text{if } supp(I_k, \{D_p, D_q\}) \geq \alpha_{p,q} \\
0, & \text{otherwise} 
\end{cases}
\]

such that

\[
supp(I_k, \{D_p, D_q\}) = \frac{supp(I_k, D_p) \times |D_p| + supp(I_k, D_q) \times |D_q|}{|D_p| + |D_q|}
\]

and

\[
\alpha_{p,q} = \frac{\alpha_p \times |D_p| + \alpha_q \times |D_q|}{|D_p| + |D_q|}
\]

We note that the operator \(| \cdot |\) is the cardinality of the set passed in as argument. Multiplying \( \alpha_p \) by \(|D_p|\) returns the minimum number of transactions in which a frequent itemset \( I_k \) should occur in \( D_p \). Therefore, \( \alpha_{p,q} \) is the minimum percentage of transactions from the cluster \( C_{p,q} = \{ D_p, D_q \} \) containing the itemset \( I_k \), i.e., \( supp(I_k, C_{p,q}) \geq \alpha_{p,q} \). In fact, the similarity measure \( sim \) in Formula (3) takes into account the local minimum support threshold at each database to calculate a new threshold for each cluster. In this paper, instead of writing 'the similarity measure \( sim \) in formula (3)', we often write \( sim \) (3).

3.1.2. Clustering Generation and Evaluation

Let \( C(D, \delta) = \{C_1, C_2, \ldots, C_k\} \) be a candidate clustering of \( D = \{D_0, D_1, \ldots, D_{n-1}\} \) produced at a given level of similarity \( \delta_i \in [0, 1] \), such that \( \cap_{j=1}^{k} \{C_j\} = \emptyset \) and \( \cup_{j=1}^{k} \{C_j\} = D \). From a graph-theoretic perspective, each cluster \( C_j \) represents a connected component in a similarity graph \( G = (D, E) \), and an edge \( (D_p, D_q) \) is added to the list of edges \( E \) if and only if \( sim(D_p, D_q) \geq \delta_{ij} \), where \( p = 0, \ldots, n - 2, q = p + 1, \ldots, n - 1 \).

Initially, \( G = (D, E) \) has no edge, i.e., \( E = \emptyset \). Then, at a given similarity level \( \delta_i \in [0, 1] \), edges \( (D_p, D_q) \) satisfying \( sim(D_p, D_q) \geq \delta_{ij} \) are added to \( E \). The level of similarity \( \delta_i \) \( (i = 1, \ldots, m) \) is chosen from the list of the \( m \) unique sorted pairwise similarities \( sim(D_p, D_q) \) computed between the \( n \) transaction databases, such that \( \delta_1 > \delta_2 > \cdots \delta_{i-1} > \delta_i > \delta_{i+1} > \cdots > \delta_m \) and \( m \leq (n^2 - n) / 2 \). After adding all the edges \( (D_p, D_q) \) at \( \delta_i \), each graph component \( C_j \) \( (j = 1, \ldots, k) \) will be representing one database cluster in our candidate partitioning \( C(D, \delta_i) \). One can then use one of the clustering goodness measures shown in Table 3 to assess the quality of \( C(D, \delta_i) \).

Once we generate and evaluate all the \( m \leq (n^2 - n) / 2 \) candidate clusterings, we report the global optimum (minimum or maximum) of the goodness measure and compare its corresponding clustering with the ground truth if it is known or with the clustering
generated at the maximum point of the silhouette coefficient when the ground truth is unknown. In fact, the silhouette coefficient $SC(D) \in [-1, 1]$ proposed in \cite{43,44} (See the last row in Table 3) could be used to verify the correctness of the cluster labels assigned to the $n$ transactional databases. Precisely, a value $SC(D) \approx 1$ suggests that the $n$ transactional databases are highly matched to their own clusters and loosely matched to their neighboring clusters.

We should note that each clustering goodness measure in Table 3 depends on more than two monotonic functions. For instance, the quality measure $goodness$ (see the first row in Table 3) proposed in \cite{20} is based on maximizing both the intra-cluster similarity $W(D)$ (which is a non-decreasing function on the interval $[0,1]$) and the inter-cluster distance $B(D)$ (which is a non-increasing function on the interval $[0,1]$), while minimizing the number of clusters $f(D)$ (which is a non-increasing function on the interval $[0,1]$). Consequently, as it was shown via the experiments done in \cite{25,26}, most of the time, the graphs of the objectives functions in Table 3 show a non-convex behavior, which makes identifying the ideal partitioning a hard problem to solve without generating and evaluating all the candidate clusterings generated at the local optima.
Table 3. A summary of the clustering quality measures mentioned in this paper.

| Clustering Quality [Reference] | Function (Equation) | Optimal Value |
|--------------------------------|---------------------|---------------|
| **goodness(\(\mathcal{D}\)) = B(\(\mathcal{D}\)) + W(\(\mathcal{D}\)) - f(\(\mathcal{D}\))** |
| \[ B(\mathcal{D}) = \sum_{C_i, C_j \in \mathcal{C}, i < j} \sum_{D_p \in C_i, D_q \in C_j, |p| < |q|} (1 - \text{sim}(D_p, D_q)) \] |
| \[ W(\mathcal{D}) = \sum_{C_i \in \mathcal{C}} \sum_{D_p, D_q \in C_i, |p| < |q|} \text{sim}(D_p, D_q) \times 1 \{(D_p, D_q) \in E\} \] |
| \[ f(\mathcal{D}) : \text{number of clusters.} \] |
| **max goodness(\(\mathcal{D}\))** |
| **goodness^2(\(\mathcal{D}\)) = \frac{\text{sum-dist}(\(\mathcal{D}\)) + \text{coupling}(\(\mathcal{D}\)) + f(\(\mathcal{D}\)) - 1}{n-1}** |
| \[ \text{sum-dist}(\mathcal{D}) = \sum_{C_i \in \mathcal{C}} \sum_{D_p, D_q \in C_i, |p| < |q|} (1 - \text{sim}(D_p, D_q)) \times 1 \{(D_p, D_q) \in E\} \] |
| \[ \text{coupling}(\mathcal{D}) = \sum_{C_i, C_j \in \mathcal{C}, i < j} \sum_{D_p \in C_i, D_q \in C_j, |p| < |q|} \text{sim}(D_p, D_q) \] |
| **min goodness^2(\(\mathcal{D}\))** |
| **goodness^3(\(\mathcal{D}\)) = \frac{\text{intra-sim}(\(\mathcal{D}\)) + \text{inter-dist}(\(\mathcal{D}\))}{f(\(\mathcal{D}\))}** |
| \[ \text{intra-sim}(\mathcal{D}) = \frac{1}{n |\mathcal{D}|^2} \sum_{C_i \in \mathcal{C}} \left\{ \frac{1}{|C_i|} \sum_{D_p \in C_i} \text{sim}(D_p, D_q) \times 1 \{(D_p, D_q) \in E\}, |C_i| > 1 \right\} \] |
| \[ \text{inter-dist}(\mathcal{D}) = \left\{ \begin{array}{ll} 0, f(\mathcal{D}) = 1 & \frac{1}{n |\mathcal{D}|^2} \times \sum_{D_p \in C_i, D_q \in C_j, |p| < |q|} (1 - \text{sim}(D_p, D_q)) \times 1 \{(D_p, D_q) \in E\}, f(\mathcal{D}) > 1 \end{array} \right\} \] |
| **max goodness^3(\(\mathcal{D}\))** |
| **SC(\(\mathcal{D}\)) = \frac{1}{n} \sum_{p=1}^{n-1} s(D_p)** |
| \[ s(D_p) = \left\{ \begin{array}{ll} \frac{b(D_p) - a(D_p)}{\max\{a(D_p), b(D_p)\}}, & |C_p| > 1; \\ 0, & \text{if } |C_p| = 1 \end{array} \right\} \] |
| \[ a(D_p) = \sum_{D_q \in C_p, p < q} (1 - \text{sim}(D_p, D_q)) \times 1 \{(D_p, D_q) \in E\} \] |
| \[ b(D_p) = \min_{D_q \in C_q} \frac{1}{|C_q|} \sum_{D_q \in C_q} (1 - \text{sim}(D_p, D_q)) \] |
| **max SC(\(\mathcal{D}\))** |
3.2. Similarity Matrix Fuzziness Reduction

In this subsection, we present our fuzziness reduction model applied to the pairwise similarities between \( n \) multiple databases. Let \( z_{pq} = \theta_{pq} \times x_{pq} \) be a weighted similarity, such that \( x_{pq} = \text{sim}(D_p, D_q) \) is the similarity value between \( D_p \) and \( D_q \) using Formula (3) and \( \theta_{pq} \) is the weight value associated with \( x_{pq} \) where \( p = 0, \ldots, n-2, q = p+1, \ldots, n-1 \). Let \( g : \mathbb{R} \to [0,1] \) be a continuous piecewise linear activation function and \( \frac{dg}{dz_{pq}} \) be its partial derivative defined as follows:

\[
g(z_{pq}, \epsilon) = \max(z_{pq}, \epsilon) - \frac{\text{sgn}(z_{pq} - 1 + \epsilon) + 1}{2} (z_{pq} - 1 + \epsilon) \tag{7}
\]

\[
\frac{dg(z_{pq}, \epsilon)}{dz_{pq}} = \frac{\text{sgn}(z_{pq} - \epsilon) + 1}{2} - \frac{\text{sgn}(z_{pq} - 1 + \epsilon) + 1}{2} \tag{8}
\]

The graph plots of \( g(z_{pq}, \epsilon) \) and \( \frac{dg(z_{pq}, \epsilon)}{dz_{pq}} \) with respect to \( z_{pq} \) are depicted in Figure 1a. The parameter \( \epsilon \) ensures that each value \( z_{pq} \) is within the range \([\epsilon, 1 - \epsilon]\) such that \( \epsilon \) is a very small number (e.g., \( \epsilon = 1\times10^{-7} \)) forcing \( g(z_{pq}, \epsilon) \) to be always above 0 and below 1, so that it can be plugged into our log-based loss function defined in (10).

3.2.1. Fuzziness Index

The fuzziness index of the pairwise similarity vector \( X^T = [\text{sim}(D_0, D_1), \text{sim}(D_0, D_2), \ldots, \text{sim}(D_{n-2}, D_{n-1})] \), also known as the entropy of the fuzzy set \( X^T \) [45], and defined from \( \mathbb{R}^2 \) to \([0,1]\), is given as follows:

\[
\text{Fuzziness}(X) = -\frac{2}{n^2 - n} \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} \left( \text{X}^T \cdot \log_2(X) + (1 - \text{X}^T) \cdot \log_2(1 - X) \right) = -\frac{2}{n^2 - n} \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} \left( \text{sim}(D_p, D_q) \cdot \log_2(\text{sim}(D_p, D_q)) + (1 - \text{sim}(D_p, D_q)) \cdot \log_2(1 - \text{sim}(D_p, D_q)) \right) \tag{9}
\]

The smaller the value of \( \text{Fuzziness}(X) \), the better the clustering performance, and vice versa. In fact, reducing the fuzziness of the pairwise similarities will lead to a more crisp decision making when it comes to finding the optimal partitioning of the \( n \) multiple databases. Particularly, the fuzziness of the similarity matrix increases when the pairwise values are centered around 0.5, resulting in more confusion when we need to decide whether two databases should be in the same cluster or not.

3.2.2. Proposed Model and Algorithm

To reduce the fuzziness associated with the \((n^2 - n)/2\) pairwise similarities between the \( n \) transaction databases \( D = \{D_0, D_1, \ldots, D_{n-1}\} \), we need to make the similarity values that are above the mean value \( \mu = 0.5 \) go closer to 1, and adjust the similarity values that are below \( \mu = 0.5 \) to go closer to 0. To do so, we consider the minimization of the sum of the binary entropy loss functions over the \((n^2 - n)/2\) weighted similarity values \( z_{pq} = \theta_{pq} \times x_{pq} \) as follows:

\[
\arg \min_\theta \mathcal{H}(\theta, \epsilon) = \arg \min_\theta \frac{2}{n^2 - n} \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} \mathcal{H}(g(z_{pq}, \epsilon))
\]

\[
= \arg \min_\theta \frac{2}{n^2 - n} \sum_{p=0}^{n-2} \sum_{q=p+1}^{n-1} \left( g(z_{pq}, \epsilon) \cdot \log_2(g(z_{pq}, \epsilon)) + (1 - g(z_{pq}, \epsilon)) \cdot \log_2(1 - g(z_{pq}, \epsilon)) \right)
\]

\[
= \arg \min_\theta \frac{2}{n^2 - n} \left( g(\theta^T \odot X^T, \epsilon) \cdot \log_2(g(\theta^T \odot X^T, \epsilon)) + (1 - g(\theta^T \odot X^T, \epsilon)) \cdot \log_2(1 - g(\theta^T \odot X^T, \epsilon)) \right)
\]

such that \( n \) is the number of databases, \( \theta^T = [\theta_{0,1}, \theta_{0,2}, \ldots, \theta_{n-2,n-1}] \) represents the model weight vector, \( z_{pq} \) represents the weighted similarity \( \theta_{pq} \times \text{sim}(D_p, D_q) \) and \( g(z_{pq}, \epsilon) \) is the activation function defined in (7). The graph plots of \( \mathcal{H}(g(z_{pq}, \epsilon)) \) and
\[
\frac{\partial H(g(z_{p,q}, \epsilon))}{\partial g(z_{p,q}, \epsilon)} \text{ with respect to } g(z_{p,q}, \epsilon) \text{ are depicted in Figure 1b. Since the fuzziness of the similarity matrix is influenced by the weights associated with the pairwise similarities, the degree to which a pair of databases } (\mathcal{D}_p, \mathcal{D}_q) \text{ belongs to the same cluster could be changed by adjusting the corresponding weight } \theta_{p,q}, \text{ which is learned by minimizing (10) via gradient descent and back-propagation. The training equations are derived as follows:}
\]

\[
\theta_{p,q} = \theta_{p,q} - \eta \frac{\partial H(\theta, \epsilon)}{\partial \theta_{p,q}}
\]

where

\[
\frac{\partial H(\theta, \epsilon)}{\partial \theta_{p,q}} = \frac{-2}{n^2 - n} \frac{\partial g(z_{p,q}, \epsilon)}{\partial z_{p,q}} \log_2 \left( \frac{g(z_{p,q}, \epsilon)}{1 - g(z_{p,q}, \epsilon)} \right)
\]

\[= \frac{-2}{n^2 - n} \frac{\partial g(z_{p,q}, \epsilon)}{\partial z_{p,q}} x_{p,q} \log_2 \left( \frac{g(z_{p,q}, \epsilon)}{1 - g(z_{p,q}, \epsilon)} \right) \tag{12}\]

Let \( \eta_0 \) and \( \text{epochs} \) be the initial learning rate and the maximum number of learning iterations, respectively. At each epoch \( i \), the current learning rate \( \eta \) decreases as follows:

\[
\eta = \eta_0 \times (1 - i / \text{epochs}) \tag{13}\]

We note that selecting a large learning rate value may cause global minimum over-shooting, whereas choosing a small learning rate may necessitate many iterations for the algorithm to converge. Hence, it is reasonable to let the learning rate decrease over time as the algorithm converges to the global minimum. In Figure 2 and Algorithm 1, we present in detail, the framework and the algorithm of the proposed fuzziness reduction model. The proposed learning Algorithm 1: SimFuzzinessReduction keeps adjusting the weight vector \( \theta \) by moving in the opposite direction to the gradient of the loss function \( H(\theta, \epsilon) \) until it reaches the maximum number of iteration \( \text{epochs} \) or until the magnitude of the gradient vector becomes below the minimum value \( \epsilon \). After convergence, we can feed the new similarity values \( [\log_2 \left( g(\theta_0 \times \text{sim}(\mathcal{D}_0, \mathcal{D}_1), \epsilon), g(\theta_0 \times \text{sim}(\mathcal{D}_0, \mathcal{D}_2), \epsilon), \ldots, g(\theta_0 \times \text{sim}(\mathcal{D}_{n-2}, \mathcal{D}_{n-1}), \epsilon) \right)] = g(\theta \odot X, \epsilon) \) to any similarity-based clustering algorithm in order to improve the quality of the produced clustering when the latter is trivial or irrelevant.

**Algorithm 1: SimFuzzinessReduction**

**Data:** \( n \): number of databases,
\[X^T = [\text{sim}(\mathcal{D}_0, \mathcal{D}_1), \text{sim}(\mathcal{D}_0, \mathcal{D}_2), \ldots, \text{sim}(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})]: \text{pairwise similarities}\]

**Parameters:** \( \theta^T = [\theta_{0,1}, \theta_{0,2}, \ldots, \theta_{n-2,n-1}]: \text{weight vector}\)

**Hyper-parameters:** \( \text{epochs} \): max number of learning cycles, \( \eta_0 \): initial learning rate

**Result:** \( g(\theta_0 \times \text{sim}(\mathcal{D}_0, \mathcal{D}_1), \epsilon), g(\theta_0 \times \text{sim}(\mathcal{D}_0, \mathcal{D}_2), \epsilon), \ldots, g(\theta_0 \times \text{sim}(\mathcal{D}_{n-2}, \mathcal{D}_{n-1}), \epsilon) \) = \( g(\theta \odot X, \epsilon) \) \( \triangleq \) adjusted pairwise similarities generating the minimum fuzziness index

1 begin
2 \( \theta^{(0)} \leftarrow \tilde{T} = [1, 1, \ldots, 1] \) \( \triangleright \) Initialize the weight parameters at iteration 0
3 \( i \leftarrow 1 \) \( \triangleright \) Initialize the iteration counter
4 \( \epsilon \leftarrow 1 \times 10^{-7} \) \( \triangleright \) Initialize \( \epsilon > 0 \), a positive small number which prevents passing a zero value to \( \log_2 \) function
5 while \( i < \text{epochs} \land \| \nabla_{g^{(i-1)}} H(\theta^{(i-1)}, \epsilon) \| > \epsilon \) do
6 \( \eta \leftarrow \eta_0 \times (1 - (i - 1) / \text{epochs}) \) \( \triangleright \) Update the learning rate
7 \( \theta^{(i)} \leftarrow \theta^{(i-1)} - \eta \nabla_{g^{(i-1)}} H(\theta^{(i-1)}, \epsilon) \) \( \triangleright \) Update the weight vector using (11) (12)
8 \( i \leftarrow i + 1 \)
9 return \( g(\theta \odot X, \epsilon) \)

Figure 1. (a): represents (in green) the graph of the piecewise linear activation function \( g(\cdot) \) and (in red) its partial derivative. We note that \( z_{p,q} = \theta_p \cdot x_{p,q} \), and \( \theta_p \) is the weight associated with the similarity value \( x_{p,q} = \text{sim}(D_p, D_q) \), \( \text{sgn} : \mathbb{R} \rightarrow \{-1, 1\} \) is the signum function and \( e \) is a small number \( \approx 1e-7 \) ensuring that \( g(z_{p,q}, e) \) is always above 0 and below 1. (b): represents the binary entropy function \( H : (0,1) \rightarrow (0,1) \) in blue and its partial derivative in orange.

Figure 2. Proposed fuzziness reduction model on the \( (n^2 - n)/2 \) pairwise similarities \( x_{p,q} = \text{sim}(D_p, D_q) \), \( p = 0, \ldots, n - 2 \), \( q = p + 1, \ldots, n - 1 \). We note that the graphs corresponding to the activation function \( g(\cdot) \) and the binary entropy function \( H(\cdot) \) are plotted in Figure 1.
3.3. Proposed Coordinate Descent-based Clustering

In this subsection, we present and discuss our proposed loss function and our coordinate descent-based clustering approach in detail. Unlike the gradient-based clustering in [25,26], our algorithm is learning-rate-free and needs to run at most $(n^2 - n)/2$ learning cycles to converge to the global minimum, such that $n$ is the number of transaction databases. In fact, at each iteration, the largest coordinate variable $\theta_{p,q}$ is selected and popped from a max-heap data structure (initially built by pushing the $(n^2 - n)/2$ pairwise similarities onto the heap). Then, we minimize our quadratic convex hinge-based loss $L(\theta)$ over $\theta_{p,q}$ which is then adjusted by moving in the opposite direction to the gradient of $L(\theta)$. This process continues until satisfying a convergence test, which will be defined later in this subsection. Each bloc of selected coordinate variables $\theta_{p,q}$ that have the same value will form a set of edges to be added to our graph $G = (D,E)$. Determining the disjoint connected components in $G$ after convergence will allow us to discover the optimal database clusters maximizing the intra-cluster similarity and the inter-cluster distance.

3.3.1. Proposed Loss Function and Algorithm

In order to implement our coordinate descent-based clustering, we propose a quadratic version of the hinge loss $L(\theta)$:

$$L(\theta) = \arg\min_{\theta} \sum_{r=0}^{n-2} \sum_{s=r+1}^{n-1} \frac{1}{2} \max(0, 1 - g(\theta_{r,s}))^2$$

A simplified 3D graph plot of $L(\theta)$ is depicted in Figure 3.

![Figure 3](image-url)

**Figure 3.** A simplified 3D plot of our proposed loss function $L(\theta)$ as defined in (14), where $\theta = [\theta_1, \theta_2]$ for visualization purposes. $P_1, P_2, P_3, P_4, A, B, C$ are some selected 3D points at which $L(\theta)$ is evaluated. From $P_1$ all the way down to $P_4$, we can clearly see that $L(\theta)$ decreases monotonically when the coordinate variables $\theta_1$ and $\theta_2$ increase their values. That is, $\forall (\theta_{i-1}^1, \theta_{i-1}^2, \theta_{i-1}^3, \theta_{i-1}^4) \in \mathbb{R}^4, \theta_{i-1}^1 \geq \theta_i^1 \land \theta_{i-1}^2 \geq \theta_i^2 \land L(\theta_{i-1}^1, \theta_{i-1}^2) \leq L(\theta_i^1, \theta_i^2)$, where $i$ is an integer representing the current iteration in our algorithm.
Initially, the weight vector $\theta^T$ is set to the $\binom{n}{2}$ pairwise similarities $X^T = [\text{sim}(D_0, D_1), \text{sim}(D_0, D_2), \ldots, \text{sim}(D_{n-2}, D_{n-1})]$, and then each weight component of $\theta^T$ is pushed onto a max-heap data structure. At each iteration $i = 1, \ldots, \binom{n}{2}$, the weight $\theta_{p,q}^{(i)}$ ($p = 0, \ldots, n - 2, q = p + 1, \ldots, n - 1$) associated with the current largest similarity value $\text{sim}(D_p, D_q)$ is popped from the max-heap and is updated as follows:

$$
\theta_{p,q}^{(i)} = \theta_{p,q}^{(i-1)} - \eta \frac{\partial \mathcal{L}(\theta^{(i-1)})}{\partial \theta_{p,q}^{(i-1)}} \\
= \theta_{p,q}^{(i-1)} + \eta (1 - \mathcal{g}^{(i)}(\theta_{p,q}^{(i-1)})) \frac{\partial \mathcal{g}^{(i)}(\theta_{p,q}^{(i-1)})}{\partial \theta_{p,q}^{(i-1)}}
$$

(15)

Such that $\mathcal{g} : \mathbb{R} \to [0, 1]$ is a differentiable activation function defined as follows:

$$
\mathcal{g}^{(i)}(\theta_{p,q}) = \max(\theta_{p,q}, 0) - \frac{\text{sgn}(\theta_{p,q} - 1) + 1}{2} \times (\theta_{p,q} - 1)
$$

(16)

and its partial derivative with respect to the weight $\theta_{p,q}$ is:

$$
\frac{\partial \mathcal{g}^{(i)}(\theta_{p,q})}{\partial \theta_{p,q}} = \frac{\text{sgn}(\theta_{p,q}) + 1}{2} - \frac{\text{sgn}(\theta_{p,q} - 1) + 1}{2}
$$

(17)

We note that $\text{sgn} : \mathbb{R} \to \{-1, 1\}$ is the signum function. The usage of $\mathcal{g}^{(i)}(\cdot)$ ensures that each weight $\theta_{p,q}$ is within the range $[0, 1]$. As there is no learning rate and schedule to choose for our coordinate descent-based algorithm, we set $\eta$ to 1.

**Theorem 1.** $\mathcal{L}(\theta)$ (14) is convex satisfying the following inequality [46]:

$$
\mathcal{L}((1-\varepsilon)\theta^{(i+1)} + \varepsilon \theta^{(i)}) \leq (1-\varepsilon)\mathcal{L}(\theta^{(i+1)}) + \varepsilon \mathcal{L}(\theta^{(i)})
$$

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

**Proof.** To prove the convexity of $\mathcal{L}(\theta)$, we can show that its Hessian matrix $\mathbf{H}$ $\mathcal{L}$ is positive semi-definite as follows:

$$
\mathbf{H} \mathcal{L} = \frac{\partial^2 \mathcal{L}}{\partial \theta_{p,q} \partial \theta_{r,s}} = 
\begin{bmatrix}
\frac{\partial^2 \mathcal{L}}{\partial \theta_{0,0} \partial \theta_{0,0}} = 1, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{0,0} \partial \theta_{0,1}} = 0, \ldots, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{0,0} \partial \theta_{2,2,n-1}} = 0 \\
\frac{\partial^2 \mathcal{L}}{\partial \theta_{0,1} \partial \theta_{0,0}} = 1, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{0,1} \partial \theta_{0,1}} = 0, \ldots, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{0,1} \partial \theta_{2,2,n-1}} = 0 \\
\vdots & \vdots & \ddots \\
\frac{\partial^2 \mathcal{L}}{\partial \theta_{2,2,n-1} \partial \theta_{0,0}} = 0, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{2,2,n-1} \partial \theta_{0,1}} = 0, \ldots, & \frac{\partial^2 \mathcal{L}}{\partial \theta_{2,2,n-1} \partial \theta_{2,2,n-1}} = 1
\end{bmatrix}
$$

Since $\mathbf{H}$ is positive semi-definite satisfying $\mathbf{x}^T \mathbf{H} \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^{\binom{n}{2}}$, $\mathcal{L}(\theta)$ is convex, and therefore guarantees convergence to the global minimum.

In order to reach the global minimum of $\mathcal{L}(\theta)$ (i.e., $\min \mathcal{L}(\theta) = 0$), our learning algorithm needs to set the weight vector $\theta$ to $\mathbf{1}$ (i.e., the unit vector). Consequently, the intra-cluster similarity will reach its maximum value and all the $n$ databases will be put into the same cluster resulting in a meaningless partitioning. Therefore, in order to prevent this scenario from occurring, we need to assess the clustering quality after popping all the coordinate variables that have the same weight $\theta_{p,q}$ (i.e., a block of weights having the same value) from the max-heap. This corresponds to generating one candidate clustering by adding the list of edges $(D_p, D_q)$ satisfying $\text{sim}(D_p, D_q) \geq \theta_{p,q}$ to the graph $G = (D, E)$. Afterward, we need a stopping condition to terminate our algorithm if the current candidate clustering quality is judged to be the optimal one in terms of
the similarity-intra cluster \( W_{\theta(i)}(D) \) and the number of clusters \( f_{\theta(i)}(D) \). For this purpose, we define the following quasi-convex loss function evaluated at the \( i \)-th iteration:

\[
L(\theta(i)) = \frac{1}{2} (f_{\theta(i)}(D) - W_{\theta(i)}(D))^2 \\
= \frac{1}{2} (f_{\theta(i)}(D) - \varphi(\theta(i))^T \cdot X)^2 \\
= \frac{1}{2} (f_{\theta(i)}(D) - \sum_{p=0}^{n-2} \sum_{\sim p+1} \text{sim}(D_p, D_q) \times \varphi(\theta(i))_{pq})^2
\]

(19)

where \( \varphi : \mathbb{R}^{(2)} \rightarrow \{0,1\}^{(2)}, \varphi(\theta) = \frac{\text{sign}(\theta - 1/2)}{2} \).

Our algorithm terminates right after it reaches the global minimum of \( L(\cdot) \). In other words, if \( L(\theta(i)) \leq L(\theta(i-1)) \), then we continue updating the weight vector, the clustering labels and save the optimal partitioning found so far. Otherwise, the algorithm terminates as it has reached the global minimum \( L(\theta(i-1)) \), and therefore, the optimal partitioning saved so far is returned as the ideal clustering of the \( n \) transactional databases. This stopping condition is only possible due to the quasi-convexity of \( L(\cdot) \).

**Theorem 2.** \( L(\theta) \) (19) is quasi-convex satisfying the following inequality [46]:

\[
L((1-\epsilon)\theta^{(i+1)} + \epsilon\theta^{(i)}) \leq \max\{L(\theta^{(i+1)}), L(\theta^{(i)})\}
\]

for all \( \theta^{(i+1)}, \theta^{(i)} \in \mathbb{R}^{(2)} \) with \( \epsilon \in [0,1] \)

**Proof.** To prove the quasi-convexity of \( L(\theta) \), we need to demonstrate the validity of (20). First, since \( f_{\theta}(D) \) is a decreasing function on the range \([0,1]\), it is then both quasi-concave and quasi-convex satisfying the following: \( f((1-\epsilon)\theta^{(i+1)} + \epsilon\theta^{(i)}) \leq \max\{f(\theta^{(i+1)}), f(\theta^{(i)})\} \) for all \( \theta^{(i+1)}, \theta^{(i)} \in \mathbb{R}^{(2)} \) with \( \epsilon \in [0,1] \). Since \( W_{\theta}(D) \) is an increasing function on the range \([0,1]\), it is then both quasi-concave and quasi-convex satisfying the following: \( W((1-\epsilon)\theta^{(i+1)} + \epsilon\theta^{(i)}) \geq \min\{W(\theta^{(i+1)}), W(\theta^{(i)})\} \) for all \( \theta^{(i+1)}, \theta^{(i)} \in \mathbb{R}^{(2)} \) with \( \epsilon \in [0,1] \). By subtracting the two last inequalities, we obtain: \( f((1-\epsilon)\theta^{(i+1)} + \epsilon\theta^{(i)}) - W((1-\epsilon)\theta^{(i+1)} + \epsilon\theta^{(i)}) \leq (\max\{f(\theta^{(i+1)}), f(\theta^{(i)})\} - \min\{W(\theta^{(i+1)}), W(\theta^{(i)})\}) \). Since \( f(\theta^{(i+1)}) \leq f(\theta(i)) \) and \( W(\theta^{(i+1)}) \geq W(\theta(i)) \), the right side of the resulting inequality is equal to \( f(\theta(i)) - W(\theta(i)) \), which could be set equal to \( \max\{f(\theta^{(i+1)}), f(\theta^{(i)})\} - W(\theta(i)) \). Finally, by squaring and dividing both sides of the inequality by \( 2 \), we get a variation on the Jensen inequality for quasi-convex functions [46] as defined in (20). Hence, \( L(\theta) \) is quasi-convex. \( \square \)

3.3.2. Time Complexity Analysis

In this subsection, we analyze the time complexity of our coordinate descent-based clustering algorithm presented in Algorithm 2, named CDCClustering, which depends on the two subroutines presented in Algorithm 3: union and Algorithm 4: cluster. We note that the superscript \( i \) enclosed in round brackets, i.e., \( \theta_{p,q}^{(i)} \), is used to indicate the iteration number at which a given variable \( \theta_{p,q} \) has been assigned a value. The proposed algorithm takes as argument the \( \binom{n}{2} \) pairwise similarities \( X = [\text{sim}(D_0, D_1), \text{sim}(D_0, D_2), \ldots, \text{sim}(D_{n-2}, D_{n-1})] \) and outputs the optimal clustering minimizing our proposed loss function \( \mathcal{L}(\theta) \) (14). First, the weight vector \( \theta^T \) is initially set equal to \( X^T \). Afterward, coordinate descent and back-propagation are used to search for the optimal weight vector \( \theta^T \), which minimizes our hinge-based objective \( \mathcal{L}(\theta) \). Through each learning cycle \( i \), one coordinate variable \( \theta_{p,q} \) is popped from a max-heap. Then, \( \theta_{p,q} \) is updated by making the optimal step in the opposite direction to the gradient of \( \mathcal{L}(\theta) \). The weights \( \theta_{p,q} (p = 0, \ldots, n-2, q = p+1, \ldots, n-1) \) attaining the maximum value of 1 will have their corresponding database pairs \( (D_p, D_q) \) put into the same cluster. By using a max-heap data structure within our coordinate descent algorithm, we optimally choose
the current largest variable $\theta_{ij}^{(i)}$ at each iteration $i$ such that taking the partial derivative of our loss $L(\theta)$ with respect to $\theta_{ij}$ allows us to attain the next steepest descent minimizing $L(\theta)$ without using a learning rate. This way, the maximum number of iterations required for our algorithm to converge is less than or equal to $(n^2 - n)/2$, i.e., the number of the pairwise similarities. Initially, the number of clusters $f_\theta(D)$ is set equal to the number of transactional databases $n$. Then, in order to keep track of the database clusters, their number $f_\theta(D)$ and their sizes, we implement a disjoint-set data structure [47], which consists of an array $A[0, \ldots, n - 1]$ of $n$ integers managed by two main operations: cluster and union. Each cluster $C_p$ is represented by a tree whose root index $p$ satisfies $A[p] = -1$, and a database $D_q$ belonging to the cluster $C_p$ satisfies $A[q] = p$. Therefore, the cluster function is called recursively to find the label assigned to the database index $p$ (passed in as argument) by moving down the tree towards the root (i.e., $A[p] = -1$). On the other hand, the union procedure links two disjoint clusters $C_p$ and $C_q$ by making the root of the smaller tree point to the root of the larger one in $A[0, \ldots, n - 1]$. The algorithms corresponding to union and cluster are presented in Algorithm 3 and Algorithm 4, respectively. Let $s = \binom{n}{2}$ be the size of the weight vector $\theta^T$. The time complexity of building the max-heap is $O(s)$ and the time complexity of the proposed Algorithm 2: CDClustering is $O(s + h \log_2(n))$, such that $h \in [1, s]$ is the number of learning cycles run until global minimum convergence, and $O(\log_2(n))$ is the time complexity of one pop operation from the heap. The proposed model is also illustrated in Figure 4. Since it is meaningless to return a single cluster consisting of all the $n$ databases, if the clustering obtained at step (10a) is trivial (i.e., all the $n$ databases are put together in one class or each single database stands alone in its own cluster), then we first need to run the model proposed in Figure 2 on the pairwise similarities to reduce the associated intrinsic fuzziness measured in (9). Afterward, we can apply the proposed model Figure 4 on the new adjusted similarity values to obtain more relevant results.
Algorithm 2: CDClustering

Data: \( n \): number of databases,
\[
X^T = \{\text{sim}(\mathcal{D}_0, \mathcal{D}_1), \text{sim}(\mathcal{D}_0, \mathcal{D}_2), \ldots, \text{sim}(\mathcal{D}_{n-2}, \mathcal{D}_{n-1})\}; \text{pairwise similarity values}
\]

Result: \( A_{best}[0, \ldots, n-1] \): cluster labels assigned to the \( n \) databases, \( f_{\theta(i)}(\mathcal{D}) \): number of clusters.

1 begin
2 \( \theta^{(0)} \leftarrow X \) \footnote{Initialize the weight parameters at iteration 0}
3 build a max-heap from the weight components \( \theta^{(0)}_{p,q} \), for \( p = 0, \ldots, n-2 \), \( q = p+1, \ldots, n-1 \)
4 for \( p \leftarrow 0 \) to \( n-1 \) do (A\([p],A_{best}\([p]\)) \leftarrow (-1, -1) \footnote{Initialize the labels assigned to the \( n \) databases}
5 \( f_{\theta^{(0)}}(\mathcal{D}) \leftarrow n \) \footnote{Initialize the number of clusters}
6 \( W_{\theta^{(0)}}(\mathcal{D}) \leftarrow X^T \cdot \phi(\theta^{(0)}) \) \footnote{Initialize the intra-graph similarity}
7 \( L(\theta^{(0)}) \leftarrow (0.5) \times (f_{\theta^{(0)}}(\mathcal{D}) - W_{\theta^{(0)}}(\mathcal{D}))^2 \) \footnote{Compute the convergence test function at iteration 0}
8 \( i \leftarrow 1 \) \footnote{Initialize the iteration counter}
9 while heap is not empty \& \( f_{\theta^{(i-1)}}(\mathcal{D}) > 1 \) do
10 \( \theta^{(i-1)}_{p,q} \leftarrow \text{pop from heap} \footnote{Get the weight variable associated with the current largest similarity}
11 \( \theta^{(i)}_{p,q} \leftarrow \theta^{(i-1)}_{p,q} + (1 - g(\theta^{(i-1)}_{p,q})) \cdot \frac{\partial g(\theta^{(i-1)}_{p,q})}{\partial \theta^{(i-1)}_{p,q}} \) \footnote{Update the weight variable at the current \( (p,q) \) coordinates}
12 \( C_p \leftarrow \text{cluster}(p, A) \) \footnote{Get the cluster label assigned to \( \mathcal{D}_p \)}
13 \( C_q \leftarrow \text{cluster}(q, A) \) \footnote{Get the cluster label assigned to \( \mathcal{D}_q \)}
14 if \( C_p \neq C_q \) then
15 union(\( C_p, C_q, A \)) \footnote{Cluster merging: link the smaller cluster to the larger one}
16 \( f_{\theta^{(i)}}(\mathcal{D}) \leftarrow f_{\theta^{(i-1)}}(\mathcal{D}) - 1 \footnote{Decrement the number of clusters after a union operation}
17 \) else if peek max from heap \( \neq \theta^{(i-1)}_{p,q} \) then
18 \( W_{\theta^{(i)}}(\mathcal{D}) \leftarrow X^T \cdot \phi(\theta^{(i)}) \) \footnote{Compute the intra-graph similarity at iteration \( i \)}
19 \( L(\theta^{(i)}) \leftarrow (0.5) \times (f_{\theta^{(i)}}(\mathcal{D}) - W_{\theta^{(i)}}(\mathcal{D}))^2 \) \footnote{Compute the convergence test function at iteration \( i \)}
20 \) else if \( L(\theta^{(i)}) \leq L(\theta^{(i-1)}) \) then
21 \( A_{best}[0, \ldots, n-1] \leftarrow A[0, \ldots, n-1] \footnote{Store the current clustering}
22 else break while loop \footnote{Algorithm terminates after satisfying the global optimality criterion}
23 \( i \leftarrow i + 1 \)
24 return(\( A_{best}, f_{\theta^{(i-1)}}(\mathcal{D}) \))
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

begin
    if |A[C_p]| > |A[C_q]| then
        A[C_p] ← A[C_p] + A[C_q] ▷ sum up the two cluster sizes
        A[C_q] ← C_p ▷ establish a link to the root of the larger cluster
    end
    else
        A[C_q] ← A[C_q] + A[C_p]
        A[C_p] ← C_q
    end
end

Algorithm 4: cluster

Data: A[0, ..., n-1]: Array representing the n cluster labels
      p: the index of D_p in A[0, ..., n-1]

Result: The cluster label of D_p in A[0, ..., n-1]

begin
    if A[p] ≥ 0 then
        ▷ If p is not a root node then move down the tree towards the root
        A[p] ← cluster(A[p], A)
        return A[p]
    end
    else
        return p ▷ return the root
    end
end
Figure 4. The coordinate descent-based clustering model depicted in eleven steps.
4. Performance Evaluation

To assess the performance of the proposed clustering algorithm, we carried out numerous experiments on real and synthetic datasets, including Zoo [48], Iris [48], Mushroom [48] and T10I4D100K [49]. To simulate a multi-database environment, we have partitioned each dataset horizontally into \( n \) partitions \( D_1, D_2, \ldots, D_n \), such that \( n \in \{12, 10, 6, 4\} \). Afterward, given a minimum support threshold \( \alpha \in \{0.5, 0.2, 0.03\} \), we run FP-Growth [1] on each partition \( D_i \) (\( i = 1, \ldots, n \)) to discover the local frequent itemsets (FIs) corresponding to each partition. All the details related to the partition sizes and their corresponding FIs are shown in Table A1. We note that the fifth column of Table A1 reports the number of FIs discovered in the entire dataset, whereas the most right column of the same table reports the number of FIs aggregated from the local FIs mined from the partitions in each cluster.

The proposed similarity measure \( \text{sim} (3) \) is called on the \( (n^2 - n) / 2 \) pairs of FIs to compute the \( n \times n \) similarity matrices shown in Figures A1a–A7a. Next, using the obtained pairwise similarities, candidate clusterings are produced via the process described in Section 3.1.2, and then evaluated using the clustering quality measures defined in Table 3, including \( SC(D) \) [43], \( \text{goodness}^3(D) \) [21], \( \text{goodness}^2(D) \) [23], \( \text{goodness}(D) \) [20] and our proposed loss function \( L(\theta) \) (14). The graphs corresponding to the studied goodness measures are shown in Figures A1b–A7b, where the optimal point (maximum or minimum) of each objective function is depicted as a black dot on its corresponding graph, except that for the graph of our loss function \( L(\theta) \), there is a red dot representing the value \( L(\arg \min \theta) \) (i.e., the optimal point at which our algorithm terminates). It is worth mentioning that due to scale differences, we sometimes multiply or divide our loss function \( L(\theta) \) (i.e., the optimal point at which our algorithm terminates). The experimental results depicted in Figures A1–A7 are summarized in Table A2, such that \( \delta \in [0, 1] \) is the ideal similarity threshold for which a goodness measure attains its optimal point. Python version 3.9.2 was used to implement all the algorithms, and the codes were run on a Ubuntu-20.04 server equipped with an Intel(R) Xeon(R) CPU clocked at 2.30 GHz with 50 GB available Disk capacity and 12 GB of available RAM.

4.1. Similarity Accuracy Analysis

To demonstrate the efficiency of \( \text{sim} (3) \), let us have three transaction databases, \(|D_1|=200\), \(|D_2|=300\) and \(|D_3|=200\) with their corresponding local frequent itemsets: \( \text{FIS}(D_1, 0.2)=\{\langle C, 0.2 \rangle, \langle B, 0.2 \rangle, \langle A, 0.2 \rangle\} \), \( \text{FIS}(D_2, 0.15)=\{\langle E, 0.9 \rangle, \langle C, 0.2 \rangle, \langle B, 0.2 \rangle, \langle A, 0.2 \rangle\} \) and \( \text{FIS}(D_3, 0.25)=\{\langle E, 0.9 \rangle\} \) mined at different minimum support threshold values \( \alpha_1 = 0.2, \alpha_2 = 0.15 \) and \( \alpha_3 = 0.25 \), respectively. Now, clustering the three databases using the algorithm BestDatabaseClustering [22] equipped with two different similarity measures, \( \text{simi} \) proposed in [20] and our proposed similarity measure \( \text{sim} (3) \), shows the results reported in Table 4. We note that \( \text{goodness} \) [20] is a clustering quality measure, such that the higher the value of \( \text{goodness} \) for a given candidate clustering \( C \), the better the quality of \( C \).

| Output | Clustering 1 under \( \text{simi} \) [20] | Clustering 2 under \( \text{sim} (3) \) |
|--------|-------------------------------|-------------------------------|
| clusters | \{\( D_1 \]\}, \{\( D_2 \), \( D_3 \)\} | \{\( D_1 \), \( D_2 \)\}, \{\( D_3 \)\} |
| Similarity intra-cluster | 0.6 | 0.75 |
| Distance inter-cluster | 1.6 | 1.75 |
| Measure goodness [20] | 0.2 | 0.5 |
As we can see, after reducing the fuzziness index associated with the previous similarity measure estimates the patterns post-mined from each cluster produce meaningful non-trivial clusterings with an increase in the silhouette coefficient. To demonstrate the importance of reducing the fuzziness associated with a similar-Ity matrices in Figures 5a and 6a, the algorithm BestDatabaseClustering [22] was able to run BestDatabaseClustering [22] on the new similarity matrices and show the clustering results in Figures 7b,c and 8b,c corresponding to rows 3 and 4 of Table 6, respectively. Afterward, we can see, after reducing the fuzziness index associated with the previous similarity matrices in Figures 5a and 6a, the algorithm BestDatabaseClustering [22] was able to produce meaningful non-trivial clusterings with an increase in the silhouette coefficient SC(D) [43] for both similarity matrices in Figures 7a and 8a.

From Table 4, we notice that using our similarity measure sim (3), we have obtained a larger intra-cluster similarity, a larger inter-cluster distance and a larger goodness [20].

| Synthesized Itemsets $I_k$ | $supp(I_k, C_{2,3})$ under simi [20] | $supp(I_k, C_{1,2})$ under sim (3) |
|---------------------------|--------------------------------------|-------------------------------------|
| A                         | 0.12 < $a_{2,3} = 0.19$              | 0.2 > $a_{1,2} = 0.17$              |
| B                         | 0.12 < $a_{2,3} = 0.19$              | 0.2 > $a_{1,2} = 0.17$              |
| C                         | 0.12 < $a_{2,3} = 0.19$              | 0.2 > $a_{1,2} = 0.17$              |
| E                         | 0.9 > $a_{2,3} = 0.19$              | 0.54 > $a_{1,2} = 0.17$              |
Table 6. A summary of the results obtained in Figures 5–8. We note that $\delta_{opt}$ is the optimal similarity level at which $\text{goodness}(D)$ [20] attains its maximum value, and $\theta^T$ is the optimal weight vector learned after a number of epochs.

| Similarity Matrix | Fuzziness Index (9) | $\theta^T, \text{epochs, } \eta$ | max $\text{goodness}(D)$ [20] | $\delta_{opt}$ | $\text{SC}(D)$ [43,44] at $\delta_{opt}$ | Optimal Clustering at $\delta_{opt}$ |
|-------------------|---------------------|---------------------------------|----------------|----------------|-----------------------------------|----------------------------------|
| Figure 5          | 0.97                | $\theta^T = [1, 1, \ldots, 1]$ (Without fuzzness reduction) | 4.19          | 0.46           | −1                                | $\{D_1, D_2, D_3, D_4, D_5\}$ |
| Figure 6          | 0.95                | $\theta^T = [1, 1, \ldots, 1]$ (Without fuzzness reduction) | 1.29          | 0.313          | −1                                | $\{D_1, D_2, D_3, D_4\}$ |
| Figure 7          | 0.74                | $\theta^T = [1.30, 0.52, 0.71, 0.71, 0.52, 0.71, 0.71, 0.52, 0.52, 1.44]$, $\text{epochs} = 300$, $\eta = 0.1$ | 4.54          | 0.95           | 0.73                              | $\{D_1, D_2\}, \{D_3\}, \{D_4, D_5\}$ |
| Figure 8          | 0.81                | $\theta^T = [0.63, 0.638, 0.591, 0.712, 0.712, 0.77]$, $\text{epochs} = 100$, $\eta = 0.1$ | 1.27          | 0.292          | 0.08                              | $\{D_1, D_2, D_3\}, \{D_4\}$ |

Figure 5. (a) A $5 \times 5$ similarity matrix between five transactional databases before applying our fuzziness reduction model. (b) Represents the graph plots corresponding to $\text{goodness}(D)$ [20], the silhouette coefficient [43] and the number of clusters. (c) Represents the optimal graph obtained at $\max \text{goodness}(D)$ [20].

Figure 6. (a) Represents a similarity matrix between four databases partitioned from the Mushroom dataset [48]. We note that (a) is built by calling $\text{sim}(\alpha)$ on the frequent itemsets (FIs) mined from $D_p$ ($p = 1, \ldots, 4$) under a threshold $\alpha = 0.5$. (b) Represents the graph plots corresponding to $\text{goodness}(D)$ [20], the silhouette coefficient [43] and the number of clusters. (c) Represents the optimal graph obtained at $\max \text{goodness}(D)$ [20].
Number of clusters

(a) Similarity matrix with a fuzziness index=0.74

(b) Clustering quality measures

(c) Optimal graph $G = (D, E)$ at $\delta = 0.95$, corresponding to $\text{max } \text{goodness}(D) = 4.54$, $\text{sim}_{\text{intra}}(D) = 1.9$, $\text{dist}_{\text{inter}}(D) = 5.64$

Figure 7. (a) A $5 \times 5$ similarity matrix obtained after applying our fuzziness reduction model on Figure 5 (a). (b) Represents the graph plots corresponding to $\text{goodness}(D)$ [20], the silhouette coefficient [43] and the number of clusters. (c) Represents the optimal graph obtained at $\text{max } \text{goodness}(D)$ [20].

Figure 8. (a) Represents the similarity table generated after applying our fuzziness reduction model on Figure 6a. (b) Represents the graph plots corresponding to $\text{goodness}(D)$ [20], the silhouette coefficient [43] and the number of clusters. (c) Represents the optimal graph obtained at $\text{max } \text{goodness}(D)$ [20].

4.3. Convexity and Clustering Analysis

In this part of our experiments, we analyze the convex behavior of the proposed clustering quality functions $L(\theta)$ (19) and $\mathcal{L}(\theta)$ (14), and we also examine the non-convexity of the existing goodness measures in [20,21,23,43]. Additionally, we compare the clustering produced by our algorithm and the ones generated at the optimal points of the previous compared goodness measures (i.e., at $\text{max } \text{goodness}(D)$ [20], $\text{min } \text{goodness}^2(D)$ [23] and $\text{max } \text{goodness}^3(D)$ [21]) with the underlying ground-truth cluster labels. When the actual clustering is unknown, we replace it with the partitioning obtained at the maximum value of the silhouette coefficient [43], that is, at $\text{max } \text{SC}(D)$. All the graphs corresponding to our loss functions and the compared goodness measures in Table 3 are plotted in Figures A1b–A7b, where the $x$-axis represents the similarity levels $\delta$ at which multiple candidate clusterings are generated and evaluated.

Consider the $7 \times 7$ similarity matrix shown in Figure A1a. From the graphs plotted in Figure A1b and according to the results shown in the first row of Table A2, we can see that using our loss function $\mathcal{L}(\theta)$ and $\text{goodness}(D)$ [20], we were able to find the optimal clustering $\{C_1 = \{D_3, D_2, D_1\}, C_2 = \{D_4\}, C_3 = \{D_7, D_6, D_5\}\}$ at a similarity level $\delta = 0.44$ where the silhouette coefficient reaches its maximum value $\text{SC}(D) = 0.46$. On the other
hand, goodness\(^3\)\((\mathcal{D})\) \([21]\) and goodness\(^2\)\((\mathcal{D})\) \([23]\) did not successfully discover the partitioning maximizing the silhouette coefficient. Additionally, we observe that the proposed convergence test function \(L(\theta)\) has a quasi-convex behavior (see proof of Theorem 2). This allows us to terminate the clustering process right after reaching the global minimum. Conversely, the graphs corresponding to goodness\(^2\)\((\mathcal{D})\) \([23]\) and goodness\((\mathcal{D})\) \([20]\) have local optima. Consequently, it is required to explore about \((n^2 - n)/2\) similarity levels in order to generate and evaluate all the candidate clusterings possible.

Now, let us examine the results of some experiments that we have conducted on the synthetic and real-world datasets shown in Table A1. From Figures A7b and A2b (the last and second rows of Table A2), we observe that goodness\(^3\)\((\mathcal{D})\) \([21]\) and goodness\(^2\)\((\mathcal{D})\) \([23]\) attain their optimal values when all the partition databases are clustered together in one class. The same phenomenon is observed in Figures A7b, A6b and A3b (the last, the sixth and the third rows of Table A2), where both goodness\(^2\)\((\mathcal{D})\) \([23]\) and goodness\((\mathcal{D})\) \([20]\) have put all the databases into one cluster.

In contrast, the proposed loss function \(L(\theta)\) has successfully identified the clustering for which the silhouette coefficient SC is maximum. Precisely, in Figure A7b, which corresponds to the last row of Table A2, \(L(\theta)\) was the only clustering quality measure which has properly identified the ideal 7-class clustering at \(\delta = 0.846\).

From the obtained graphs in Figures A1–A7, we notice that goodness\(^3\)\((\mathcal{D})\) \([21]\), goodness\(^2\)\((\mathcal{D})\) \([23]\) and goodness\((\mathcal{D})\) \([20]\) are neither quasi-concave nor quasi-convex on the domain \([0, 1]\). As a result, we observe the existence of local optimum points on their corresponding graphs, which makes the search of the global optimum a difficult problem to solve without exploring all the local solutions.

Conversely, we observe that our loss function \(L(\theta)\) \((14)\) is monotonically decreasing all the time and \(\mathcal{L}(\hat{\theta}) = 0\) at \(\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta)\). This corresponds to the similarity level \(\delta = 0\) where all the \(n\) databases are put into the same single cluster. To avoid this case from occurring, we used the quasi-convex function \(L(\theta)\) \((19)\) as a convergence test function to terminate our algorithm at the point \(L(\arg\min_{\theta} \mathcal{L}(\theta))\) corresponding to the red dot on the graph of our loss function \(L(\theta)\). Moreover, it is worth noting that for every two real \((\frac{1}{2}\mathcal{L}\)\)-dimensional vectors \(\theta^{(i)}\) and \(\theta^{(i+1)}\), where \(L(\theta^{(i+1)}) \leq L(\theta^{(i)})\), the line that joins the points \((\theta^{(i+1)}, L(\theta^{(i)}))\) and \((\theta^{(i)}, L(\theta^{(i)}))\) remains above \(L(\theta)\), which is observed in Figures A1–A7. Therefore, using the proposed loss function \(L(\theta)\) \((14)\) along with \(L(\theta)\) \((19)\) guarantees global minimum convergence.

In the fifth and the most right columns of Table A1, we compare the number of frequent itemsets (FIs) mined from all the partitions of a given dataset \(\mathcal{D}\) with the FIs mined from each single cluster \(C_j\) consisting of similar partitions from the same dataset, where \(\bigcup_{j=1}^{\mathcal{C}} \{C_j\} = \emptyset\) and \(\bigcup_{j=1}^{\mathcal{C}} \{C_j\} = \mathcal{D}\). We notice that mining all the partitions from datasets Iris \([48]\) and Zoo \([48]\) did not result in discovering any valid frequent itemset. Whereas, mining each individual cluster of partitions from the datasets Iris and Zoo has led to the discovery of new patterns in each cluster \(C_j\).

In Table A3, we report the similarity levels \(\delta_{opt}\) at which the clustering evaluation measures goodness\((\mathcal{D})\) \([20]\), goodness\(^2\)\((\mathcal{D})\) \([23]\), goodness\(^3\)\((\mathcal{D})\) \([21]\), the silhouette coefficient SC \([43]\) and our proposed loss function \(L(\theta)\) attain their optimal values in Figures A1–A7. We note that, the fraction \([\delta_{opt} - \delta_{stop}]\) \([\delta_{opt} - \delta_{stop}]\) in Table A3 represents the number of similarity levels required to test the convergence and terminate divided by the number of all similarity levels \(m\). We note that \(stop = opt < m\). On the other hand, as for the compared algorithms, which are based on non-convex objectives, we notice that \(stop = m\). Therefore, our algorithm requires the least number of similarity levels \((opt \text{ out of } m)\) in order to converge and terminate, which makes our algorithm faster than the compared algorithms in \([21–23]\), requiring to generate and evaluate all the \(m\) candidate clusterings in order to return the optimal one.

All the previous results confirm that using our loss function \(L(\theta)\) \((14)\) along with \(L(\theta)\) \((19)\), we have identified the ideal clustering for which the silhouette coefficient SC \([43]\)
is maximum and we have also improved the quality of the frequent itemsets (FIs) mined from the multiple databases partitioned from the datasets in Table A1.

4.4. Clustering Error and Running Time Analysis

In this experimental part, we compare the running time of the proposed clustering algorithm with the execution times of two clustering algorithms for multi-database mining (MDM), namely GDMDBClustering [25] and BestDatabaseClustering [22], all run on the same random data samples. We also calculate how the clusterings produced by our algorithm and the compared models are different from the ground-truth clustering. For this purpose, we propose an error function in (21), which measures the difference between two given clusterings $P$ and $Q$.

First, we generated $n=30$ to $N=120$ isotropic Gaussian blobs using the scikit-learn generator [50], such that the number of features for each $n$ blobs is set to $\text{random.randint}(2, 10)$, while the number of clusters is set to $\lfloor \frac{n}{2} \rfloor$. In Table 7, we present a brief summary of the random blobs generated via scikit-learn [50].

| Number of Random Blobs $(n)$ | Number of Centers $\lfloor \frac{n}{2} \rfloor$ | Number of Attributes $(m)$ |
|-----------------------------|---------------------------------|-----------------------------|
| 30                          | 15                              | random.randint(2, 10)       |
| ...                         | ...                             | ...                         |
| 60                          | 30                              | random.randint(2, 10)       |
| ...                         | ...                             | ...                         |
| 120                         | 60                              | random.randint(2, 10)       |

Afterward, we use the min-max scaling [51] to normalize each feature (out of the $m$ features) into the interval $[0,1]$. Then, for each $n$ blobs, every pair of $m$-dimensional blobs is passed as arguments to the function $\text{sim}(3)$ in order to compute the $\frac{(n^2 - n)}{2}$ pairwise similarities between the $n$ blobs. We then run the proposed algorithm, GDMDBClustering [25] (with three different learning rate values) and BestDatabaseClustering [22] on each of the $\frac{(n^2 - n)}{2}$ pairwise similarities ($n = 30, \ldots, 120$) and plot their running time graphs in Figures A8a–A10a, and then plot the clustering error graphs in Figures A8b–A10b.

Without loss of generality, assume $Q$ is the ground-truth clustering (i.e., the actual clusters) of the current $n$ blobs $D = \{D_1, D_2, \ldots, D_n\}$ generated via scikit-learn [50], and assume $P$ is the partitioning of $D$ produced by a given clustering algorithm. To measure how far is $P$ from $Q$, we propose the error function $E_n(P, Q) \in [0,1]$ to be defined as follows:

$$E_n(P, Q) = \frac{|\text{Pairs}_Q \setminus \text{Pairs}_P| + |\text{Pairs}_P \setminus \text{Pairs}_Q|}{|\text{Pairs}_Q| + |\text{Pairs}_P|}$$

(21)

where $|\text{Pairs}_P|$ is the number of all the database pairs obtained from every cluster in $P$ and $|\text{Pairs}_P \setminus \text{Pairs}_Q|$ is the number of all the database pairs that only exist in $\text{Pairs}_P$ and that cannot be found in $\text{Pairs}_Q$. We note that $E_n(P, Q)$ approaches the maximum value of 1 (i.e., $E_n(P, Q) \approx 1$) when $P$ and $Q$ are too different and do not share many database pairs in common (i.e., $|\text{Pairs}_P \cap \text{Pairs}_Q| \approx 0$). Conversely, $E_n(P, Q) \approx 0$ when the clustering $P$ and $Q$ are too similar, i.e., they share the maximum number of pairs $(D_p, D_q)$.

We also define the average of the $N - n + 1$ clustering errors, which could also be seen as the mean absolute clustering error:

$$\overline{E}(P, Q) = \frac{\sum_{n=1}^{N} E_n(P, Q)}{N - n + 1}$$

(22)
From the obtained results in Figures A8a–A10a, we observe a rapid increase in the running time of BestDatabaseClustering [22] as the number of generated blobs \((n)\) increases linearly. This is due to the fact that BestDatabaseClustering needs to generate and evaluate approximately \((n^2 - n)/2\) candidate clusterings in order to find the optimal clustering for which the non-convex function \(goodness(D)\) [20] is maximum. In fact, \(goodness(D)\) suffers from the existence of local maxima, which requires exploring all the local candidate solutions in order to find the global maximum. On the other hand, using the proposed convex loss function \(L(\theta)\) and the quasi-convex convergence test function \(L(\theta)\) allows us to stop the clustering process at \(L(\arg \min_{\theta} L(\theta))\). Consequently, this avoids generating unnecessary candidate clusterings, and hence reduces the CPU overhead. Since our algorithm is independent of the learning rate \(\eta\), the running time of our algorithm is the same in all Figures A8a–A10a. Whereas, the running time of GDMDBClustering [25] increases for smaller learning rate values (e.g., Figure A10) and decreases when we set larger learning rate values (e.g., Figure A9), but this comes at the cost of having an increased clustering error.

Next, by examining the three clustering error graphs in Figures A8–A10 (b), we observe that BestDatabaseClustering [22] has the largest clustering error among the three algorithms with a clustering average error \(E(P, Q) = 0.936\). In fact, on average, BestDatabaseClustering [22] tends to group all the current \(n\) blobs \((n = 30, \ldots, 120)\) in one single cluster. On the other hand, our proposed algorithm and GDMDBClustering [25] produce clusterings that are close to the ground-truth clustering predetermined by the scikit-learn generator [50]. In fact, the average clustering error due to our algorithm is \(E(P, Q) = 0.285\). For GDMDBClustering [25], we get \(E(P, Q) = 0.285\) when the learning rate \(\eta = 0.0005\) or \(\eta = 0.001\), and the error increases to \(E(P, Q) = 0.29\) when \(\eta = 0.002\). The average running times and clustering errors of our algorithm, GDMDBClustering [25] and BestDatabaseClustering [22] are summarized in Table A4.

Our algorithm and GDMDBClustering [25] terminate once we reach the global minimum of the convergence test function \(L(\theta)\). Consequently, the running times of our algorithm and GDMDBClustering [25] are most of the time shorter than that of BestDatabaseClustering [22]. Overall, the running time of GDMDBClustering [25] stays relatively steady with respect to \(n\). However, GDMDBClustering depends strongly on the learning step size \(\eta\) and its decay rate. On the other hand, our algorithm is learning-rate-free and needs at most (in the worst case) \((n^2 - n)/2\) iterations to converge. Consequently, our proposed algorithm is faster than both BestDatabaseClustering [22] and GDMDBClustering [25].

To illustrate the statistically significant superiority of the proposed clustering model in terms of running time and clustering accuracy, we have applied the Friedman test [52] (under a significance level \(\alpha = 0.05\)) on the measurements (execution times and clustering errors depicted in Figures A8–A10) obtained by our algorithm, BestDatabaseClustering [22] and GDMDBClustering [25] (with three different values for the learning rate \(\eta\)) considering all the random samples in Table 7.

After conducting the Friedman test [52], we obtained the results shown in Tables A5–A7, namely the average running time, the average clustering error \(E(P, Q)\) (22), the standard deviation (SD), the variance (Var), the critical value (stat) and its \(p\)-value for all the tested clustering algorithms, considering all 91 random samples generated via scikit-learn [50].

We notice that all the obtained results in Tables A5–A7 show \(p\)-values that are below the significance level \(\alpha = 0.05\). Consequently, the test suggests a rejection of the null hypothesis, stating that the compared clustering models have a similar performance. In fact, the proposed clustering algorithm significantly outperforms the other compared models, as it has the shortest average running time (6.367 milliseconds) and the lowest average clustering error \((E(P, Q) = 0.285)\) among all the compared models.
4.5. Clustering Comparison and Assessment

In the third part of our experiments, we are interested in using some information retrieval measures to compare the clusterings produced by our algorithm and some other clustering algorithms with the ground-truth data.

Let \( D = \{D_1, D_2, \ldots, D_n\} \) be \( n \) transactional databases. Let \( P = \{P_1, P_2, \ldots, P_k\} \) be a \( k \)-class clustering of \( D \) produced by any given clustering algorithm, and let \( Q = \{Q_1, Q_2, \ldots, Q_l\} \) be the ground-truth clustering of the databases in \( D \), such that \( \bigcup_{i=1}^k \{P_i\} = \bigcup_{i=1}^l \{Q_i\} = D \) and \( \bigcap_{i=1}^k \{P_i\} = \bigcap_{i=1}^l \{Q_i\} = \emptyset \). Let us define \( Pairs_P \) and \( Pairs_Q \) as the set of database pairs obtained from each cluster of the same clustering. That is, \( Pairs_P = \bigcup_{P \in P} \bigcup_{D_r, D_s \in P \cap P_{ri < s}} \{(D_r, D_s)\} \) and \( Pairs_Q = \bigcup_{Q \in Q} \bigcup_{D_r, D_s \in Q \cap Q_{ri < s}} \{(D_r, D_s)\} \). To compare the two clusterings \( P \) with \( Q \), few methods method [53–55] could be used. In this paper, we use a pair counting [56–59] to calculate some information retrieval measures [60,61], including precision, recall, F-measure (i.e., harmonic mean of recall and precision), Rand index [62] and Jaccard index [63] over pairs of databases being clustered together in \( P \) and/or \( Q \). This will allow us to assess whether the predicted database pairs from \( P \) cluster together in \( Q \), i.e., are the discovered database pairs in \( Pairs_P \) correct with respect to the underlying true pairs in \( Pairs_Q \) from the ground-truth clustering \( Q \).

In Table A9, we show the categories of database pairs which represent the working set of all pair counting measures cited in Table A10. Precisely, \( a \): represents the number of pairs that exist in both clusterings \( Q \) and \( P \), \( d \): represents the number of pairs that do not exist in either clustering, \( b \) is the number of pairs present only in clustering \( Q \), and \( c \) is the number of pairs present only in clustering \( P \). By counting the pairs in each category, we get an indicator for agreement and disagreement of the two clusterings being compared. The following example illustrates how to compute the measures defined in Table A10 for two given clusterings \( P = \{\{D_1, D_2, D_3\}, \{D_4, D_5, D_6, D_7\}\} \) and the ground-truth partitioning \( Q = \{\{D_1, D_2, D_3\}, \{D_4\}, \{D_5, D_6, D_7\}\} \) of seven transaction databases \( D = \bigcup_{i=1}^7 \{D_i\} \). First, let us calculate the following pairing categories:

\[
Pairs_D = \bigcup_{r=1}^6 \bigcup_{s=r+1}^7 \{(D_r, D_s)\},
\]
\[
Pairs_Q = \{(D_6, D_7), (D_5, D_7), (D_5, D_6), (D_2, D_3), (D_1, D_3), (D_1, D_2)\},
\]
\[
Pairs_P = \{(D_6, D_7), (D_5, D_7), (D_5, D_6), (D_4, D_7), (D_4, D_6), (D_4, D_5), (D_2, D_3), (D_1, D_3), (D_1, D_2)\}.
\]

Then, \( a = |Pairs_Q \cap Pairs_P| = 6, b = |Pairs_Q \setminus Pairs_P| = 0, c = |Pairs_P \setminus Pairs_Q| = 3, d = |Pairs_P \setminus (Pairs_Q \cup Pairs_P)| = 12 \). Therefore, we get the following measures: \( F\text{-measure} = 0.8, \text{precision} = 0.66, \text{recall} = 1.0, \text{Rand} = 0.857, \text{Jaccard} = 0.66 \). We note that the higher the values of the evaluation measures given in Table A10, the better the matching of the clustering \( P \) to its corresponding ground-truth clustering \( Q \).

In Tables A8 A11, we report the F-measure [60,61], precision [60,61], recall [60,61], Rand [62] and Jaccard [63] reached by the clustering algorithms in [21–23], and our proposed algorithm on the datasets shown in Figures A1–A7. From Table A8 and Table A11, we notice that our algorithm achieves the best scores against the compared clustering algorithms, considering all the experiments in Figures A1–A7.

5. Conclusions

An improved similarity-based clustering algorithm for multi-database mining was proposed in this paper. Unlike the previous works, our algorithm requires fewer upper-bounded iterations to minimize a convex clustering quality measure. In addition, we have proposed a preprocessing layer prior to clustering where the pairwise similarities between multiple databases are first adjusted to reduce their fuzziness. This will help the clustering process to be more precise and less confused in discriminating between the different database clusters. To assess the performance of our algorithm, we conducted several experiments on real and synthetic datasets. Compared to the existing clustering algorithms for multi-database mining, our algorithm achieved the best performance in terms of
accuracy and running time. In this paper, we have used the most frequent itemsets mined from each transaction database as feature sets to compute the pairwise similarities between the multiple databases. However, when the sizes of these input vectors become large, building the similarity matrix will increase the CPU overhead drastically. Moreover, the existence of some noisy frequent itemsets (FIs) may largely influence how databases are clustered together. In future work, we will investigate the impact of compressing the size of the FIs into a latent variable represented in a lower dimensional space with discriminative features. Practically, reconstituting the input vectors from the embedding space using deep auto-encoders and non-linear dimensionality reduction techniques, such as T-SNE (t-distributed stochastic neighbor embedding) and UMAP (uniform manifold approximation and projection), will force the removal of the noisy features present in the input data while keeping only the meaningful discriminative ones. Consequently, this may help improve the accuracy and running time of the clustering algorithm. Additionally, we are interested in exploring new ways to reduce the computational time used to calculate the similarity matrix via locality sensitive hashing (LSH) techniques, such as BagMinHash for weighted sets. These methods aim to encode the feature-set vectors into hash-code signatures in order to efficiently estimate the Jaccard similarity between the local transactional databases. Last but not least, in order to design a parallel version of the proposed algorithm, we will study and explore some high-performance computing tools, such as MapReduce and Spark, as an attempt to improve the clustering performance for multi-database mining.

Author Contributions: Conceptualization, S.M.; methodology, S.M.; software, S.M.; validation, S.M.; formal analysis, S.M.; investigation, S.M.; resources, S.M.; data curation, S.M.; writing—original draft preparation, S.M.; writing—review and editing, S.M.; visualization, S.M., W.D.; supervision, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All datasets are available at http://fimi.ua.ac.be/data/ and https://archive.ics.uci.edu/ml/datasets.

Acknowledgments: We would like to thank the anonymous reviewers for their time and their valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

| Abbreviation | Description                      |
|--------------|----------------------------------|
| FIs          | Frequent Itemsets                |
| FIM          | Frequent Itemset Mining          |
| MDB          | Multiple Databases               |
| MDM          | Multi-database Mining            |
| CD           | Coordinate Descent               |
| CL           | Competitive Learning             |
| BMU          | Best Matching Unit               |
| T-SNE        | t-Distributed Stochastic Neighbor Embedding |
| UMAP         | Uniform Manifold Approximation and Projection |
| LSH          | Locality Sensitive Hashing       |
## Appendix A

### Table A1. Description of the datasets used in our experiments with their database partitions and the number frequent itemsets (FIs) mined from each database partition $D_i$ and the number of FIs mined from each cluster $C_j$ under a threshold $\alpha$.

| Dataset Name/Ref | Number of Rows | Number of Rows in Partition $D_i$ | Number of FIS($D_i, \alpha$) from Partition $D_i$ | Number of FIS($D, \alpha$) from Dataset $D$ | Ground Truth Clustering | Number of FIS($C_j, \alpha$) from Cluster $C_j$ |
|------------------|----------------|----------------------------------|----------------------------------------|----------------------------------------|-------------------------|----------------------------------------|
| **Mushroom** [48] (2 classes) | 8124 | $|D_1| = 3916$ (C_1) | $|FIS(D_1, 0.5)| = 375$ | $|FIS(D_1, 0.5)| = 2063$ | $C_1 = \{D_1\}$ | $|FIS(C_1, 0.5)| = 375$ |
| | | $|D_2| = 1402$ (C_2) | $|FIS(D_2, 0.5)| = 32911$ | $|FIS(D_2, 0.5)| = 807$ | $C_2 = \{D_2, D_3, D_4\}$ | $|FIS(C_2, 0.5)| = 1441$ |
| **Zoo** [48] (7 classes) | 101 | $|D_1| = 20$ (C_1) | $|FIS(D_1, 0.5)| = 24383$ | $|FIS(D_1, 0.5)| = 30975$ | $C_1 = \{D_1\}$ | $|FIS(C_1, 0.5)| = 25087$ |
| | | $|D_2| = 21$ (C_1) | $|FIS(D_2, 0.5)| = 30719$ | $|FIS(D_2, 0.5)| = 23727$ | $C_2 = \{D_2\}$ | $|FIS(C_2, 0.5)| = 28671$ |
| | | $|D_3| = 10$ (C_2) | $|FIS(D_3, 0.5)| = 30719$ | $|FIS(D_3, 0.5)| = 23727$ | $C_3 = \{D_3\}$ | $|FIS(C_3, 0.5)| = 25479$ |
| | | $|D_4| = 5$ (C_3) | $|FIS(D_4, 0.5)| = 240479$ | $|FIS(D_4, 0.5)| = 65535$ | $C_4 = \{D_4\}$ | $|FIS(C_4, 0.5)| = 49151$ |
| | | $|D_5| = 6$ (C_4) | $|FIS(D_5, 0.5)| = 65535$ | $|FIS(D_5, 0.5)| = 114467$ | $C_5 = \{D_5\}$ | $|FIS(C_5, 0.5)| = 57343$ |
| | | $|D_6| = 2$ (C_5) | $|FIS(D_6, 0.5)| = 114467$ | $|FIS(D_6, 0.5)| = 98303$ | $C_6 = \{D_6, D_7\}$ | $|FIS(C_6, 0.5)| = 45955$ |
| | | $|D_7| = 3$ (C_6) | $|FIS(D_7, 0.5)| = 98303$ | $|FIS(D_7, 0.5)| = 53247$ | $C_7 = \{D_7\}$ | $|FIS(C_7, 0.5)| = 28671$ |
| **Iris** [48] (3 classes) | 150 | $|D_1| = 25$ (C_1) | $|FIS(D_1, 0.2)| = 5$ | $|FIS(D_1, 0.2)| = 6$ | $C_1 = \{D_1\}$ | $|FIS(C_1, 0.2)| = 3$ |
| | | $|D_2| = 25$ (C_1) | $|FIS(D_2, 0.2)| = 2$ | $|FIS(D_2, 0.2)| = 2$ | $C_2 = \{D_2\}$ | $|FIS(C_2, 0.2)| = 1$ |
| | | $|D_3| = 25$ (C_2) | $|FIS(D_3, 0.2)| = 2$ | $|FIS(D_3, 0.2)| = 2$ | $C_3 = \{D_3\}$ | $|FIS(C_3, 0.2)| = 2$ |
| | | $|D_4| = 25$ (C_3) | $|FIS(D_4, 0.2)| = 2$ | $|FIS(D_4, 0.2)| = 2$ | $C_4 = \{D_4\}$ | $|FIS(C_4, 0.2)| = 2$ |
| | | $|D_5| = 25$ (C_4) | $|FIS(D_5, 0.2)| = 2$ | $|FIS(D_5, 0.2)| = 2$ | $C_5 = \{D_5\}$ | $|FIS(C_5, 0.2)| = 2$ |
| | | $|D_6| = 25$ (C_5) | $|FIS(D_6, 0.2)| = 2$ | $|FIS(D_6, 0.2)| = 2$ | $C_6 = \{D_6\}$ | $|FIS(C_6, 0.2)| = 2$ |
| | | $|D_7| = 25$ (C_6) | $|FIS(D_7, 0.2)| = 2$ | $|FIS(D_7, 0.2)| = 2$ | $C_7 = \{D_7\}$ | $|FIS(C_7, 0.2)| = 2$ |
| **T104D100K** [49] (unknown classes) | 100,000 | $|D_i| = 10,000$ rows, $i = 1 \ldots 10$ | $|FIS(D_i, 0.03)| = 58$ | $|FIS(D_i, 0.03)| = 58$ | $C_1 = \{D_1\}$ | $|FIS(C_1, 0.03)| = 58$ |
| | | | $|FIS(D_i, 0.03)| = 58$ | $|FIS(D_i, 0.03)| = 58$ | $C_2 = \{D_2\}$ | $|FIS(C_2, 0.03)| = 58$ |
| | | | $|FIS(D_i, 0.03)| = 62$ | $|FIS(D_i, 0.03)| = 62$ | $C_3 = \{D_3\}$ | $|FIS(C_3, 0.03)| = 62$ |
| | | | $|FIS(D_i, 0.03)| = 62$ | $|FIS(D_i, 0.03)| = 62$ | $C_4 = \{D_4\}$ | $|FIS(C_4, 0.03)| = 62$ |
| | | | $|FIS(D_i, 0.03)| = 62$ | $|FIS(D_i, 0.03)| = 62$ | $C_5 = \{D_5\}$ | $|FIS(C_5, 0.03)| = 62$ |
| | | | $|FIS(D_i, 0.03)| = 62$ | $|FIS(D_i, 0.03)| = 62$ | $C_6 = \{D_6\}$ | $|FIS(C_6, 0.03)| = 62$ |
| | | | $|FIS(D_i, 0.03)| = 62$ | $|FIS(D_i, 0.03)| = 62$ | $C_7 = \{D_7\}$ | $|FIS(C_7, 0.03)| = 62$ |
| **Figure A1** [20] (unknown classes) | 24 | $|D_1| = 3$ | $|FIS(D_1, 0.42)| = 3$ | $|FIS(D_1, 0.42)| = 3$ | $C_1 = \{D_1\}$ | $|FIS(C_1, 0.42)| = 3$ |
| | | $|D_2| = 3$ | $|FIS(D_2, 0.42)| = 3$ | $|FIS(D_2, 0.42)| = 3$ | $C_2 = \{D_2\}$ | $|FIS(C_2, 0.42)| = 3$ |
| | | $|D_3| = 3$ | $|FIS(D_3, 0.42)| = 3$ | $|FIS(D_3, 0.42)| = 3$ | $C_3 = \{D_3\}$ | $|FIS(C_3, 0.42)| = 3$ |
| | | $|D_4| = 4$ | $|FIS(D_4, 0.42)| = 7$ | $|FIS(D_4, 0.42)| = 7$ | $C_4 = \{D_4\}$ | $|FIS(C_4, 0.42)| = 7$ |
| | | $|D_5| = 4$ | $|FIS(D_5, 0.42)| = 7$ | $|FIS(D_5, 0.42)| = 7$ | $C_5 = \{D_5\}$ | $|FIS(C_5, 0.42)| = 7$ |
| | | $|D_6| = 4$ | $|FIS(D_6, 0.42)| = 5$ | $|FIS(D_6, 0.42)| = 5$ | $C_6 = \{D_6\}$ | $|FIS(C_6, 0.42)| = 3$ |
| | | $|D_7| = 4$ | $|FIS(D_7, 0.42)| = 3$ | $|FIS(D_7, 0.42)| = 3$ | $C_7 = \{D_7\}$ | $|FIS(C_7, 0.42)| = 3$ |
Table A2. Clustering results illustrated in Figures A1–A7 after using the clustering quality measures $goodness^3(D)$ [21], $goodness^2(D)$ [23], the silhouette coefficient $SC(D)$ [43], $goodness(D)$ [20] and our proposed objective function $L(\theta)$, where $\delta$ is the level of similarity at which each clustering evaluation/loss function reaches its optimal value.

| Dataset      | Silhouette Coefficient | Clustering Result Using Proposed Objective | Clustering Result/ Optimal Value | Clustering Result/ Optimal Value | Clustering Result/ Optimal Value |
|--------------|------------------------|-------------------------------------------|--------------------------------|--------------------------------|--------------------------------|
| Name/Ref     | $\max SC(D)$          | $\min goodness^2(D)$                     | $\max goodness^3(D)$          | $\min goodness^2(D)$                     | $\max goodness^3(D)$          |
| Figure A1 7×7 [20] | 0.46 at $\delta = 0.444$ | $C_1 = \{D_{a1}, D_{a2}, D_{a3}\}$       | 2.004 at $\delta = 0.444$ | $C_1 = \{D_{b1}, D_{b2}, D_{b3}\}$       | 15.407 at $\delta = 0.444$ |
| Figure A2 12×12 [48] | 0.41 at $\delta = 0.559$ | $C_1 = \{D_{a1}, D_{a2}, D_{a3}\}$       | 7.71 at $\delta = 0.559$ | $C_1 = \{D_{b1}, D_{b2}, D_{b3}\}$       | 32.98 at $\delta = 0.559$ |
| Figure A3 4×4 Mushroom [48] | 0.08 at $\delta = 0.41$ | $C_1 = \{D_{a1}, D_{a2}\}$               | 0.43 at $\delta = 0.41$ | $C_1 = \{D_{b1}, \ldots, D_{b1}\}$       | 1.672 at $\delta = 0.365$ |
| Figure A4 6×6 Iris [48] | 0.304 at $\delta = 0.3$ | $C_1 = \{D_{a1}, D_{a2}, D_{a3}\}$       | 1.10 at $\delta = 0.3$ | $C_1 = \{D_{b1}, \ldots, D_{b1}\}$       | 9.64 at $\delta = 0.3$ |
| Figure A5 7×7 Zoo & Mushroom [48] | 0.63 at $\delta = 0.384$ | $C_1 = \{D_{a1}, D_{a2}\}$               | 0.5 at $\delta = 0.384$ | $C_1 = \{D_{b1}, \ldots, D_{b1}\}$       | 9.96 at $\delta = 0.384$ |
| Figure A6 4×4 [39] | 0.34 at $\delta = 0.429$ | $C_1 = \{D_{a1}, D_{a2}\}$               | 1.12 at $\delta = 0.429$ | $C_1 = \{D_{b1}, \ldots, D_{b1}\}$       | 2.708 at $\delta = 0.25$ |
| Figure A7 10×10 | 0.115 at $\delta = 0.846$ | $C_1 = \{D_{a1}, D_{a2}\}$               | 0.7 at $\delta = 0.846$ | $C_1 = \{D_{b1}, \ldots, D_{b1}\}$       | 35.275 at $\delta = 0.737$ |

Figure A1. (a): represents a similarity matrix between 7 databases from [25]. We note that (a) is built by calling $sim(3)$ on the frequent itemsets (FIs) mined from $D_p (p = 1, \ldots, 7)$ under a threshold $\alpha = 0.42$. (b): represents the graph plots corresponding to $goodness^3(D) \times 10$ [21], $goodness^2(D) \times 10$ [23], $goodness(D)$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta)$, our proposed loss function $L(\theta) \times 5$ and the number of generated clusters.
Figure A2. (a): represents a similarity matrix between 12 databases partitioned from Zoo dataset [48]. We note that (a) is built by calling $\text{sim}(\cdot)$ on the frequent itemsets (FIs) mined from $D_p (p=1, \ldots, 12)$ under a threshold $\alpha = 0.5$. (b): represents the graph plots corresponding to $\text{goodness}^3(\mathcal{D}) \times 10$ [21], $\text{goodness}^2(\mathcal{D}) \times 10$ [23], $\text{goodness}(\mathcal{D})$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta) \div 10$, our proposed loss function $L(\theta) \times 2$ and the number of generated clusters.

Figure A3. (a): represents a similarity matrix between 4 databases partitioned from Mushroom dataset [48] without applying the fuzziness reduction model in Figure 2. We note that (a) is built by calling $\text{sim}(\cdot)$ on the frequent itemsets (FIs) mined from $D_p (p=1, \ldots, 4)$ under a threshold $\alpha = 0.5$. (b): represents the graph plots corresponding to $\text{goodness}^3(\mathcal{D}) \times 10$ [21], $\text{goodness}^2(\mathcal{D}) \times 10$ [23], $\text{goodness}(\mathcal{D})$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta)$, our proposed loss function $L(\theta) \times 10$ and the number of generated clusters.
| Database | D1 | D2 | D3 | D4 | D5 | D6 |
|----------|----|----|----|----|----|----|
| D1       | 1  | .439 | 0  | 0  | 0  | 0  |
| D2       | .439 | 1  | 0  | 0  | 0  | 0  |
| D3       | 0  | 0   | 1  | .300 | 0  | 0  |
| D4       | 0  | 0   | .300 | 1  | .278 | .171 |
| D5       | 0  | 0   | .278 | 1  | .357 | 1  |
| D6       | 0  | 0   | .171 | .357 | 1  | 1  |

(a) Similarity matrix

Figure A4. (a): represents a similarity matrix between 6 databases partitioned from the Iris dataset [48]. We note that (a) is built by calling sim (3) on the frequent itemsets (FIs) mined from $D_p$ ($p = 1, \ldots, 6$) under a threshold $\alpha = 0.2$. (b): represents the graph plots corresponding to $\text{goodness}^3(D) \times 10$ [21], $\text{goodness}^2(D) \times 10$ [23], $\text{goodness}(D)$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta)$, our proposed loss function $L(\theta) \times 10$ and the number of generated clusters.

| Database | D1 | D2 | D3 | D4 | D5 | D6 |
|----------|----|----|----|----|----|----|
| D1       | 1  | .895 | 0  | 0  | 0  | 0  |
| D2       | .895 | 1  | 0  | 0  | 0  | 0  |
| D3       | 0  | 0   | 1  | .493 | .486 | .50 |
| D4       | 0  | 0   | .493 | 1  | .613 | .492 |
| D5       | 0  | 0   | .486 | .613 | 1  | .603 |
| D6       | 0  | 0   | .384 | .492 | .603 | 1  |

(a) Similarity matrix

Figure A5. (a): represents a similarity matrix between 6 databases including 4 databases partitioned from the real dataset Mushroom [48] and 2 databases partitioned from the real dataset Zoo [48]. We note that (a) is built by calling sim (3) on the frequent itemsets (FIs) mined from $D_p$ ($p = 1, \ldots, 6$) under a threshold $\alpha = 0.5$. (b): represents the graph plots corresponding to $\text{goodness}^3(D) \times 10$ [21], $\text{goodness}^2(D) \times 10$ [23], $\text{goodness}(D)$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta)$, our proposed loss function $L(\theta) \times 10$ and the number of generated clusters.

| Database | D1 | D2 | D3 | D4 | D5 | D6 |
|----------|----|----|----|----|----|----|
| D1       | 1  | .538 | .429 | .176 | 0  | 0  |
| D2       | .538 | 1  | .667 | .176 | 0  | 0  |
| D3       | .429 | .667 | 1  | .250 | 0  | 0  |
| D4       | 0  | .176 | .250 | 1  | 0  | 0  |

(a) Similarity matrix

Figure A6. (a): represents a similarity matrix between 4 databases from [39]. (b): represents the graph plots corresponding to $\text{goodness}^3(D) \times 10$ [21], $\text{goodness}^2(D) \times 10$ [23], $\text{goodness}(D)$ [20], the silhouette coefficient $SC \times 10$ [43], our convergence test function $L(\theta)$, our proposed loss function $L(\theta) \times 10$ and the number of generated clusters.
Figure A7. (a): represents a similarity matrix between 10 databases partitioned from T10H4D100K [49]. We note that (a) is built by calling sim (3) on the frequent itemsets (FIs) mined from \( D_p \) \((p = 1, \ldots, 10)\) under a threshold \( \alpha = 0.03\).

(b): represents the graph plots corresponding to goodness\(^3\) (\( D \)) \( \times 10\) [21], goodness\(^2\) (\( D \)) \( \times 10\) [23], goodness (\( D \)) [20], the silhouette coefficient SC \( \times 10\) [43], our convergence test function \( L(\theta) : 10\), our proposed loss function \( L(\theta) \times 10\) and the number of generated clusters.

Table A3. The similarity levels \( \delta_{opt} \) at which the clustering evaluation measures goodness\(^3\) (\( D \)) [21], goodness\(^2\) (\( D \)) [23], the silhouette coefficient SC (\( D \)) [43], goodness (\( D \)) [20] and our proposed objective function \( L(\theta) \) attain their optimal values in Figures A1–A7. The fraction \( \frac{|\{\delta_{opt} |\langle 0.41\} \rangle}{|\{\delta_{opt} \rangle\}} \) represents the number of similarity levels required to test the convergence and terminate divided by the number of total similarity levels \( m \).

| Dataset | silhouette coefficient \( \delta_{opt} \) (max SC (\( D \))) | Proposed Loss Function \( L(\arg \min \left \{ \delta_{opt} \right \} \) | Goodness Measure \( \delta_{opt} \) goodness (\( D \)) | Goodness Measure \( \delta_{opt} \) goodness (\( D \)) | Goodness Measure \( \delta_{opt} \) goodness\(^2\) (\( D \)) | Goodness Measure \( \delta_{opt} \) goodness\(^3\) (\( D \)) |
|---------|----------------------------------------------------------|----------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|
| Figure A1 7 × 7 [20] | 0.444 | 10 \( \overline{m} \) | 0.444 | 5 \( \overline{m} \) | 0.444 | 10 \( \overline{m} \) | 0.065 | 10 \( \overline{m} \) | 0.086 | 10 \( \overline{m} \) |
| Figure A2 12 × 12 Zoo [48] | 0.559 | 48 \( \overline{m} \) | 0.559 | 5 \( \overline{m} \) | 0.559 | 48 \( \overline{m} \) | 0.348 | 48 \( \overline{m} \) | 0.348 | 48 \( \overline{m} \) |
| Figure A3 4 × 4 | 0.41 | 4 \( \overline{m} \) | 0.41 | 2 \( \overline{m} \) | 0.365 | 4 \( \overline{m} \) | 0.365 | 4 \( \overline{m} \) | 0.41 | 4 \( \overline{m} \) |
| Mushroom [48] | 0.3 | 6 \( \overline{m} \) | 0.3 | 6 \( \overline{m} \) | 0.3 | 6 \( \overline{m} \) | 0.3 | 6 \( \overline{m} \) |
| Figure A5 6 × 6 Iris [48] | 0.384 | 8 \( \overline{m} \) | 0.384 | 7 \( \overline{m} \) | 0.384 | 8 \( \overline{m} \) | 0.384 | 8 \( \overline{m} \) | 0.384 | 8 \( \overline{m} \) |
| Zoo & Mushroom [48] | 0.429 | 6 \( \overline{m} \) | 0.429 | 6 \( \overline{m} \) | 0.25 | 6 \( \overline{m} \) | 0.25 | 6 \( \overline{m} \) | 0.429 | 6 \( \overline{m} \) |
| Figure A6 4 × 4 [49] | 0.846 | 31 \( \overline{m} \) | 0.846 | 4 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) |
| Figure A7 10 × 10 T10H4D100K [49] | 0.846 | 31 \( \overline{m} \) | 0.846 | 4 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) | 0.737 | 31 \( \overline{m} \) |
Table A4. Summary of the average running times and the average clustering errors $\mathcal{E}(P, Q)$ (22) for the proposed algorithm, BestDatabaseClustering [22] and GDMDBClustering [25] (with three different values for the learning rate $\eta$) on the random samples described in Table 7.

| Experiment (Figure) | Proposed Algo | BestDatabaseClustering [22] | GDMDBClustering [25] |
|---------------------|---------------|----------------------------|-----------------------|
|                     | Average Running Time | Average Clustering Error | Average Running Time | Average Clustering Error | Average Running Time | Average Clustering Error |
| Figure A8 $\eta = 0.001$ | 6.367 | 0.285 | 47.208 | 0.936 | 14.825 | 0.285 |
| Figure A9 $\eta = 0.002$ | 6.367 | 0.285 | 47.208 | 0.936 | 7.305 | 0.290 |
| Figure A10 $\eta = 0.0005$ | 6.367 | 0.285 | 47.208 | 0.936 | 28.479 | 0.285 |

Table A5. Statistical Friedman test results [52] for the measurements obtained by all the compared clustering algorithms in Figure A8.

| Algorithm | Measurements |
|-----------|--------------|
|            | Running Time | Clustering Error |
|            | Average | SD | Var | stat | p-Value | Average | SD | Var | stat | p-Value |
| Proposed Algo | 6.367 | 3.018 | 9.107 | | | |
| BestDatabaseClustering [22] | 47.208 | 27.537 | 758.313 | 135.707 | 3.40e–30 | 0.936 | 0.066 | 0.004 | 150 | 2.67e–33 |
| GDMDBClustering [25] ($\eta = 0.001$) | 14.825 | 1.743 | 3.037 | | | |

Table A6. Statistical Friedman test results [52] for the measurements obtained by all the compared clustering algorithms in Figure A9.

| Algorithm | Measurements |
|-----------|--------------|
|            | Running Time | Clustering Error |
|            | Average | SD | Var | stat | p-Value | Average | SD | Var | stat | p-Value |
| Proposed Algo | 6.367 | 3.018 | 9.107 | | | |
| BestDatabaseClustering [22] | 47.208 | 27.537 | 758.313 | 121.62 | 3.88e–27 | 0.936 | 0.066 | 0.004 | 131 | 3.56e–29 |
| GDMDBClustering [25] ($\eta = 0.002$) | 7.305 | 1.766 | 3.118 | | | |

Table A7. Statistical Friedman test results [52] for the measurements obtained by all the compared clustering algorithms in Figure A10.

| Algorithm | Measurements |
|-----------|--------------|
|            | Running Time | Clustering Error |
|            | Average | SD | Var | stat | p-Value | Average | SD | Var | stat | p-Value |
| Proposed Algo | 6.367 | 3.018 | 9.107 | | | |
| BestDatabaseClustering [22] | 47.208 | 27.537 | 758.313 | 118.90 | 1.51e–26 | 0.936 | 0.066 | 0.004 | 150 | 2.67e–33 |
| GDMDBClustering [25] ($\eta = 0.0005$) | 28.479 | 4.655 | 21.669 | | | |
Figure A8. (a): represents the running times corresponding to GDMDBClustering [25] (with a learning rate $\eta = 0.001$), BestDatabaseClustering [22] and the proposed algorithm obtained when executed on $n = 30, \ldots, 120$ isotropic Gaussian blobs generated using scikit-learn generator [50]. (b): represents the clustering error graphs (21) due to GDMDBClustering [25], BestDatabaseClustering [22] and the proposed algorithm.

Figure A9. (a): represents the running times corresponding to GDMDBClustering [25] (with a learning rate $\eta = 0.002$), BestDatabaseClustering [22] and the proposed algorithm obtained when executed on $n = 30, \ldots, 120$ isotropic Gaussian blobs generated using scikit-learn generator [50]. (b): represents the clustering error graphs (21) due to GDMDBClustering [25], BestDatabaseClustering [22] and the proposed algorithm.

Figure A10. (a): represents the running times corresponding to GDMDBClustering [25] (with a learning rate $\eta = 0.0005$), BestDatabaseClustering [22] and the proposed algorithm obtained when executed on $n = 30, \ldots, 120$ isotropic Gaussian blobs generated using scikit-learn generator [50]. (b): represents the clustering error graphs (21) due to GDMDBClustering [25], BestDatabaseClustering [22] and the proposed algorithm.
Table A8. F-measure [60,61], precision [60,61] and recall [60,61] reached by the compared clustering algorithms in [21–23], and our proposed algorithm on the datasets shown in Figures A1–A7. Notice that our algorithm gets the best scores for all the datasets.

| Dataset | F-Measure [60,61] | Precision [60,61] | Recall [60,61] |
|---------|------------------|------------------|----------------|
|         | Proposed Algo    | Algo [22]        | Algo [23]      | Algo [21]   | Proposed Algo    | Algo [22] | Algo [23] | Algo [21] |
| Figure A1 7 × 7 [20] | 1 | 1 | 0.44 | 0.8 | 1 | 1 | 0.28 | 0.66 | 1 | 1 | 1 | 1 |
| Figure A2 12 × 12 Zoo [48] | 1 | 1 | 0.14 | 0.14 | 1 | 1 | 0.075 | 0.075 | 1 | 1 | 1 | 1 |
| Figure A3 4 × 4 Mushroom [48] | 1 | 0.66 | 0.66 | 1 | 1 | 0.5 | 0.5 | 1 | 1 | 1 | 1 | 1 |
| Figure A4 6 × 6 Iris [48] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Figure A5 6 × 6 Zoo & Mushroom [48] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Figure A6 4 × 4 [39] | 1 | 0.66 | 0.66 | 1 | 1 | 0.5 | 0.5 | 1 | 1 | 1 | 1 | 1 |
| Figure A7 10 × 10 T10H3D100K [49] | 1 | 0.16 | 0.16 | 0.16 | 1 | 0.088 | 0.088 | 0.088 | 1 | 1 | 1 | 1 | 1 |

Table A9. Contingency matrix showing the categories in pairing clustered databases.

| Clustering | Predicted Clusters | Actual clusters |
|------------|--------------------|-----------------|
|            | Pairs in P         | Pairs not in P  |
| Pairs in Q | a := |Pairs_Q ∩ Pairs_P| (True Positive) |
| Pairs not in Q | b := |Pairs_Q \ Pairs_P| (False Negative) |
|             | c := |Pairs_P \ Pairs_Q| (False Positive) |
|             | d := Pairs in none| (True Negative) |

Table A10. Pair counting measures used for clustering assessment and comparison.

| Precision [60,61] | Recall [60,61] | F-Measure [60,61] | Rand [62] | Jaccard [63] |
|-------------------|----------------|-------------------|-----------|-------------|
| \( \frac{a}{(a)+(c)} \) | \( \frac{a}{(a)+(b)} \) | \( \frac{2a}{(a)+(b)+(c)} \) | \( \frac{(a)+(d)}{(a)+(b)+(c)+(d)} \) | \( \frac{a}{(a)+(b)+(c)} \) |
Table A11. Rand index [62] and Jaccard index [63] reached by the clustering algorithms in [22], [23], [21] and our proposed algorithm on the datasets shown in Figures A1–A7. Notice that our algorithm gets the best scores for all the datasets.

| Dataset       | Proposed Algo | Rand [62] | Jaccard [63] |
|---------------|---------------|-----------|--------------|
| Figure A1 7 × 7 [20] | 1 | 1 | 0.28 | 0.85 | 1 | 1 | 0.28 | 0.66 |
| Figure A2 12 × 12 Zoo [48] | 1 | 1 | 0.075 | 0.075 | 1 | 1 | 0.075 | 0.075 |
| Figure A3 4 × 4 Mushroom [48] | 1 | 0.5 | 0.5 | 1 | 1 | 0.5 | 0.5 | 1 |
| Figure A4 6 × 6 Iris [48] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Figure A5 6 × 6 Zoo & Mushroom [48] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Figure A6 4 × 4 [39] | 1 | 0.5 | 0.5 | 1 | 1 | 0.5 | 0.5 | 1 |
| Figure A7 10 × 10 T10I4D100K [49] | 1 | 0.088 | 0.088 | 0.088 | 1 | 0.088 | 0.088 | 0.088 |

References
1. Han, J.; Pei, J.; Yin, Y.; Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining Knowl. Discov.* 2004, 8, 53–87.
2. Ng, A.Y.; Jordan, M.I.; Weiss, Y. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inform. Process. Syst.* 2001, 849–856.
3. Johnson, S.C. Hierarchical clustering schemes. *Psychometrika* 1967, 32, 241–254.
4. MacQueen, J.; others. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Oakland, CA, USA, 1966, Volume 1, pp. 281–297.
5. Zhang, Y.J.; Liu, Z.Q. Self-splitting competitive learning: A new on-line clustering paradigm. *IEEE Trans. Neural Netw.* 2002, 13, 369–380.
6. Yair, E.; Zeger, K.; Gersho, A.; others. Competitive learning and soft competition for vector quantizer design. *IEEE Trans. Signal Process.* 1992, 40, 294–309.
7. Hofmann, T.; Buhmann, J.M. Competitive learning algorithms for robust vector quantization. *IEEE Trans. Signal Process.* 1998, 46, 1665–1675.
8. Kohonen, T. *Self-Organizing Maps*; Springer Science & Business Media: Berlin/Heidelberg, Germany; New York, NY, USA, 2012; Volume 30.
9. Pal, N.R.; Bezdek, J.C.; Tsao, E.K. Generalized clustering networks and Kohonen’s self-organizing scheme. *IEEE Trans. Neural Netw.* 1993, 4, 549–557.
10. Mao, J.; Jain, A.K. A self-organizing network for hyperellipsoidal clustering (HEC). *Trans. Neural Netw.* 1996, 7, 16–29.
11. Anderberg, M.R. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*; Academic Press: Cambridge, MA, USA, 2014; Volume 19.
12. Aggarwal, C.C.; Reddy, C.K. Data clustering. *Algorithms and Application*; CRC Press: Boca Raton, FL, USA, 2014.
13. Wang, C.D.; Lai, J.H.; Philip, S.Y. NEIWalk: Community discovery in dynamic content-based networks. *IEEE Trans. Knowl. Data Eng.* 2013, 26, 1734–1748.
14. Wang, Z.; Zhang, D.; Zhou, X.; Yang, D.; Yu, Z.; Yu, Z. Discovering and profiling overlapping communities in location-based social networks. *IEEE Trans. Syst. Man Cybern. Syst.* 2013, 44, 499–509.

15. Huang, D.; Lai, J.H.; Wang, C.D.; Yuen, P.C. Ensembling over-segmentations: From weak evidence to strong segmentation. *Neurocomputing* 2016, 207, 416–427.

16. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 2000, 22, 888–905.

17. Zhao, Q.; Wang, C.; Wang, P.; Zhou, M.; Jiang, C. A novel method on information recommendation via hybrid similarity. *IEEE Trans. Syst. Man Cybern. Syst.* 2016, 48, 448–459.

18. Symeonidis, P. Clustering Data: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD. *IEEE Trans. Syst. Man Cybern. Syst.* 2015, 46, 1240–1251.

19. Rafaillidis, D.; Daras, P. The TFC model: Tensor factorization and tag clustering for item recommendation in social tagging systems. *IEEE Trans. Syst. Man Cybern. Syst.* 2012, 43, 673–688.

20. Adhikari, A.; Adhikari, J. Clustering Multiple Databases Induced by Local Patterns. In *Advances in Knowledge Discovery in Databases*; Springer: Cham, Switzerland, 2015; pp. 305–332.

21. Liu, Y.; Yuan, D.; Cuan, Y. Completely Clustering for Multi-databases Mining. *J. Comput. Inform. Syst.* 2013, 9, 11144–11172, doi:10.1109/ACCESS.2021.3050404.

22. Miloudi, S.; Hebri, S.A.R.; Khiat, S. Contribution to Improve Database Classification Algorithms for Multi-Database Mining. *J. Comput. Proces. Syst.* 2018, 14, 709–726.

23. Tang, H.; Mei, Z. A Simple Methodology for Database Clustering. In Proceedings of the The 5th International Conference on Computer Engineering and Networks, SISSA Medialab, 2015; Volume 259, p. 19.

24. Wang, R.; Ji, W.; Liu, M.; Wang, X.; Weng, J.; Deng, S.; Gao, S.; Yuan, C.A. Review on mining data from multiple data sources. *Pattern Recognit. Lett.* 2018, 109, 120–128.

25. Miloudi, S.; Wang, Y.; Ding, W. A Gradient-Based Clustering for Multi-Database Mining. *IEEE Access* 2021, 9, 11144–11172, doi:10.1109/ACCESS.2021.3050404.

26. Miloudi, S.; Wang, Y.; Ding, W. An Optimized Graph-based Clustering for Multi-database Mining. In *Proceedings of the 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA*, 9–11 November 2020, pp. 807–812, doi:10.1109/ICTAI50040.2020.00128.

27. Zhang, S.; Zaki, M.J. Mining Multiple Data Sources: Local Pattern Analysis. *Data Mining Knowl. Discov.* 2006, 12, 121–125.

28. Adhikari, A.; Rao, P.R. Synthesizing heavy association rules from different real data sources. *Pattern Recognit. Lett.* 2008, 29, 59–71.

29. Adhikari, A.; Adhikari, J. Advances in Knowledge Discovery in Databases; Springer: Switzerland, 2015.

30. Adhikari, A.; Jain, L.C.; Prasad, B. A State-of-the-Art Review of Knowledge Discovery in Multiple Databases. *J. Intell. Syst.* 2017, 26, 23–34.

31. Zhang, S.; Zhang, C.; Wu, X. Identifying Exceptional Patterns. *Knowl. Discov. Multiple Datab.* 2004, pp. 185–195.

32. Zhang, S.; Zhang, C.; Wu, X. Identifying High-vote Patterns. *Knowl. Discov. Multiple Datab.* 2004, pp. 157–183.

33. Ramkumar, T.; Srinivasan, R. Modified algorithms for synthesizing high-frequency rules from different data sources. *Knowl. Inform. Syst.* 2008, 17, 313–334.

34. Djenouri, Y.; Lin, J.C.W.; Nørvåg, K.; Ramampiaro, H. Highly efficient pattern mining based on transaction decomposition. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019, pp. 1646–1649.

35. Savasere, A.; Omiecinski, E.R.; Navathe, S.B. An Efficient Algorithm for Mining Association Rules in Large Databases; Technical Report GIT-CC-95-04; Georgia Institute of Technology: 1995.

36. Zhang, S.; Wu, X. Large scale data mining based on data partitioning. *Appl. Artif. Intell.* 2001, 15, 129–139.

37. Zhang, C.; Liu, M.; NIE, W.; Zhang, S. Identifying Global Exceptional Patterns in Multi-database Mining. *IEEE Intell. Inform. Bull.* 2004, 3, 19–24.

38. Zhang, S.; Zhang, C.; Yu, J.X. An efficient strategy for mining exceptions in multi-databases. *Inform. Sci.* 2004, 165, 1–20.

39. Wu, X.; Zhang, C.; Zhang, S. Database classification for multi-database mining. *Inform. Syst.* 2005, 30, 71–88.

40. Li, H.; Hu, X.; Zhang, Y. An Improved Database Classification Algorithm for Multi-database Mining. Frontier in Algorithmics. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 346–357.

41. Na, S.; Xumin, L.; Yong, G. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In Proceedings of the 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, IEEE, Jian, China, 2–4 April 2010; pp. 63–67.

42. Selim, S.Z.; Ismail, M.A. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Intell.* 1984, 81–87.

43. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 1987, 20, 53–65.

44. Kaufman, L.; Rousseeuw, P.J. *Finding Groups in Data: An Introduction to Cluster Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 344.

45. De Luca, A.; Termini, S. A Definition of a Nonprobabilistic Entropy in the Setting of Fuzzy Sets Theory. In *Readings in Fuzzy Sets for Intelligent Systems*; Dubois, D., Prade, H., Yager, R.R., Eds.; Morgan Kaufmann: 1993; pp. 197–202. doi:10.1016/B978-1-4832-1450-4.50020-1.

46. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
47. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. Data structures for disjoint sets. In *Introduction to Algorithms*; 2009; pp. 498–524.
48. Center for Machine Learning and Intelligent Systems. UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ml/datasets/ (accessed 10 October 2020).
49. IBM Almaden Quest research group. Frequent Itemset Mining Dataset Repository. Available online: http://fimi.ua.ac.be/data/ (accessed 10 October 2020).
50. Thirion, G.; Varoquaux, A.; Gramfort, V.; Michel, O.; Grisel, G.; Louppe, J. Nothman. Scikit-learn: Sklearn.datasets.makeblobs. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html (accessed 10 October 2020).
51. Gramfort, A.; Blondel, M.; Grisel, O.; Mueller, A.; Martin, E.; Patrini, G.; Chang, E. Scikit-learn: sklearn.preprocessing.MinMaxScaler. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (accessed 10 October 2020).
52. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* 1940, 11, 86–92.
53. Meilă, M. Comparing clusterings: An axiomatic view. In Proceedings of the 22nd International Conference on Machine Learning, 2005; pp. 577–584.
54. Vinh, N.X.; Epps, J.; Bailey, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.* 2010, 11, 2837–2854.
55. Günnemann, S.; Färber, I.; Müller, E.; Assent, I.; Seidl, T. External evaluation measures for subspace clustering. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, 2011; pp. 1363–1372.
56. Banerjee, A.; Krumpelman, C.; Ghosh, J.; Basu, S.; Mooney, R.J. Model-based overlapping clustering. In Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005; pp. 532–537.
57. Pfitzner, D.; Leibbrandt, R.; Powers, D. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowl. Inform. Syst.* 2009, 19, 361–394.
58. Achtert, E.; Goldhofer, S.; Kriegel, H.P.; Schubert, E.; Zimek, A. Evaluation of clusterings–metrics and visual support. In Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, IEEE, Arlington, VA, USA, 1–5 April 2012; pp. 1285–1288.
59. Shafiei, M.; Milios, E. Model-based overlapping co-clustering. In Proceedings of the SIAM Conference on Data Mining, 2006.
60. Chinchor, N. MUC-4 evaluation metrics. In Proceedings of the of the Fourth Message Understanding Conference 22–29, 1992.
61. Mei, Q.; Radev, D. Information retrieval. In *The Oxford Handbook of Computational Linguistics*, 2nd ed.; 1979.
62. Rand, W.M. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* 1971, 66, 846–850.
63. Jaccard, P. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bull. Soc. Vaudoise Sci. Nat.* 1901, 37, 241–272.