

# Multi-type Relational Clustering Approaches: Current State-of-the-Art and New Directions

Tao Li, Sarabjot S. Anand
Department of Computer Science, University of Warwick
United Kingdom
{li.tao, s.s.anand}@warwick.ac.uk

***Abstract*—The proliferation of multi-type relational datasets in a number of important real-world applications and the limitations resulting from the transformation of such datasets to fit propositional data mining approaches have led to the emergence of the discipline of multi-type relational data mining. Clustering is an important unsupervised learning task aimed at discovering structure inherent in data. In this paper, we survey the state-of-the-art in the field of relational clustering, providing a taxonomy of approaches and review some of the most representative algorithms within each category. We also present DIVA, our general framework for multi-type relational clustering, which combines the use of Representative Objects with multi-phase clustering in a bid to provide flexibility, efficiency and effectiveness in clustering relational datasets. Theoretical analysis and experimental results prove that our approach is more effective and efficient than a number of other algorithms proposed in literature.**

***Index Terms*—Data Mining, Clustering, Algorithm, Multi-Type, Relational.**

## I. Introduction

As a widely-applied technique for data analysis and knowledge discovery, cluster analysis tries to partition a dataset into a number of finite and discrete subsets by considering the internal homogeneity and the external separation of the clusters, i.e. maximizing intra-cluster similarity and inter-cluster dissimilarity [16][43].

Traditional cluster analysis mainly focuses on propositional datasets, which are composed of a *flat* and *single-type* data structure. All data instances in a propositional dataset are of the same type with instances typically consisting of a list of numeric or nominal attribute values, so that they can be represented as points in a multi-dimensional vector space. Based on such a representation, many propositional clustering algorithms have been developed that can be classified as density-based, model-based or distance-based approaches, depending on whether they take neighbourhood characteristics into account or not. Alternatively these algorithms can be classified into partitional or hierarchical approaches, depending on whether they create a single partition of the data or a hierarchical decomposition.

A variety of algorithms for clustering propositional datasets have been proposed in literature. In particular, recent efforts focus on scalability and efficiency of clustering algorithms and their ability to discover clusters of increasingly complex shapes. For example, BIRCH [46] utilizes the concept of clustering feature (CF) to efficiently summarize the statistical characteristics of a cluster and distribute the data objects in Euclidean space into micro clusters. The CF vectors are updated when a cluster absorbs a new data object or two sub-clustered are merged. By scanning the dataset, BIRCH incrementally builds a CF-tree to preserve the inherent clustering structure of the data objects that have been scanned. Finally, in order to remedy the problems of skewed input order or undesirable splitting, BIRCH applies a traditional agglomerative clustering algorithm to improve cluster quality using the CF-tree. DBSCAN [10], a density-based clustering algorithm, aims to find clusters of arbitrary shapes. Clusters are dynamically created from an arbitrary point, absorbing all points that are reachable from within its neighbourhood (defined by two parameters: *Eps* (the radius of the neighbourhood of a point) and *MinPts* (the minimum number of points in the neighbourhood)). Chameleon [20] is a hierarchical clustering approach based on the $k$-nearest-neighbor graph. Edge are drawn between each object and its $k$ nearest neighbours. The weights assigned to these edges are the pair-wise similarities between data objects. Chameleon uses a graph-partitioning algorithm to divide the graph into a set of sub-clusters with the minimal edge cut. These sub-clusters are then repeatedly merged, by considering their *relative interconnectivity* and *relative closeness*, to derive the ultimate cluster result. More propositional clustering approaches as well as their applications can be found in the comprehensive surveys [2][18][43].

Relational datasets usually pertain to domains with a number of object types and a multitude of relationships defined between these types[7]. Data relating to these objects and relationships between objects are stored in multiple tables within a relational database. Although the propositional clustering algorithms are still applicable in these cases by means of combining the multiple tables into a single one through join or aggregation operations (a process referred to as propositionalization), it is not a good choice for the following reasons [15][31]:

- The transformation of relational linkage information into a unified feature space will causes information loss, or generate very high dimensional and sparse data, which will inevitably degrade the performance of clustering algorithms.
- The traditional clustering framework is designed only for an individual dataset, so it cannot capture the dynamic influence propagation along the paths of relations between

multi-type data.

- Besides the clusters within data objects of each type, the global hidden patterns involving multi-type objects might also be important, which cannot be recognized by classical clustering approaches.

The above limitations, of applying propositional clustering algorithms to propositionalized multi-relational data, have motivated the development of numerous algorithms for clustering multi-relational datasets. In this paper we first provide a comprehensive survey of the relational clustering approaches in Section II. After that, we present our relational clustering framework *DIVA* in Section III explaining the motivation behind various aspects of the framework. Theoretical analysis and experimental results (in Section III-D) show that DIVA is both effective and efficient. Finally, conclusions and future work on DIVA are presented in Section IV.

## II. Multi-Type Relational Clustering

Multi-type relational clustering approaches try to partition data, involving multiple tables (relations) within a relational database, into a set of clusters so as to reflect the hidden structure within the data. Here the attribute values of the data as well as the inter-relationships among them are both important for the learning procedure [8]. Generally speaking, there are three ways of designing Multi-Type Relational Data Mining (MRDM) approaches:

- Transform the MRDM problems into propositional form [27]. All related tables are firstly merged into a single one by adding new attribute-values so that traditional propositional learner can be applied. Finally, the induced hypothesis is transformed back into the relational form.
- Use the techniques of Inductive Logic Programming (ILP) to induce relational rules. The MRDM prediction problems are hence converted into the tasks of automated logic-program synthesis. It usually includes the phases of constructing clause space based on $\theta$-subsumption and performing exhaustive or heuristic search in that space [8].
- Upgrade the single table data mining algorithms to relational ones based on the observation that MRDM algorithms have many characteristics in common with propositional learning algorithms [39]. The basic idea is to keep as much of the propositional algorithm as possible and only upgrade the key notions, e.g. upgrading the distance measure and keep the clustering algorithms unchanged.

Džeroski compared the pros and cons of the above ideas [7]: The first one makes available many data mining algorithms that works on a single table, but is only feasible for a restricted class of MRDM problems. The second one is the most intuitive, but it cannot be applied to large-scale problems because ILP algorithms usually require a great deal of computational resource. Therefore, the third one is very attractive and many MRDM algorithms based on this approach have been developed recently, aimed at inducing structural regression trees [26], association rules [5], classification models [9] and clustering [22][23].

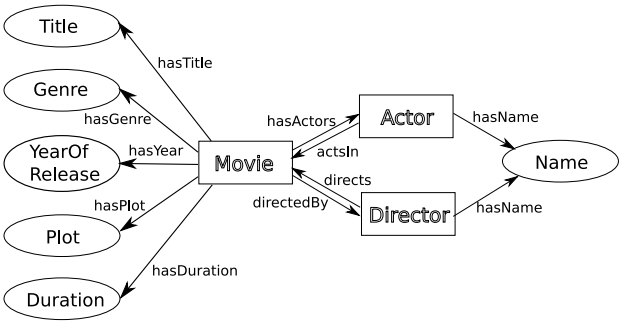


Fig. 1. Ontology of a movie dataset

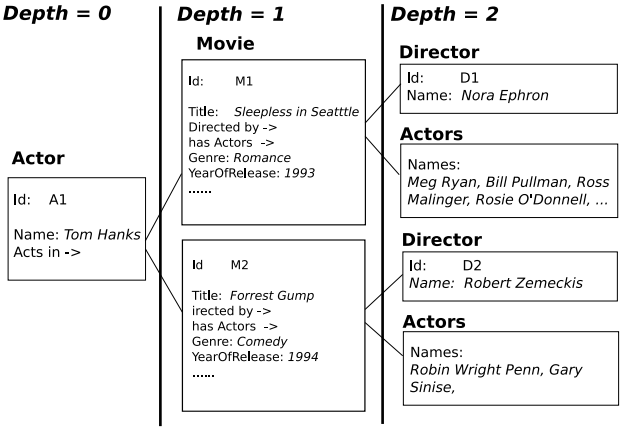


Fig. 2. Example object: Tom Hanks ($Depth = 2$)

In the following subsections, we first introduce how to construct the relational data objects and upgrade the similarity measures accordingly in Section II-A. We then provide a taxonomy of approaches to relational clustering algorithms discussing representative algorithms within each category.

### A. Relational Object Construction and Relational Similarity Measure

Given an ontology represented as a directed graph $G = (C, E)$, in which the vertices, $C$, represent the set of concepts in the ontology and edges $E = \{\vec{e}_{st} | \text{ edge } \vec{e}_{st} : c_s \rightarrow c_t;\ c_s, c_t \in C\}$ represent the relationships between concept pairs, i.e. source concept $c_s$ references target concept $c_t$ as its member property. When constructing an object $\mathbf{x}$ of concept $c_s$, we will first build its member concept list $MC(c_s)$ and then link all objects related to $\mathbf{x}$ into the member property attributes of $\mathbf{x}$. We say an object $\mathbf{y}$ of concept $c_t$ is related to $\mathbf{x}$ when $c_t \in MC(c_s)$. In such case, $\mathbf{y}$ will be added into the member property attribute $\mathbf{x}.c_t$. Then for each $\mathbf{y} \in \mathbf{x}.c_t$, we launch the above procedure iteratively until $MC(c_t) = \emptyset$ or a pre-specified depth bound $Depth(\geq 0)$ is reached. Figures 1 is the ontology of a relational movie dataset and Figures 2 shows part of the object for Tom Hanks constructed in this way.

In the propositional datasets each data point is represented as a vector in the multi-dimensional space, so it is possible to use any of the numerous approaches to calculating

dissimilarity between two data points in multi-dimensional space. Relational objects consist of multi-type components (simple types such as numeric attributes or compound types as sub-objects) and the relationships defined between different concepts. Therefore, the similarity measures for relational datasets are more complex than those for propositional ones. Emde and Wettschereck proposed the RIBL distance measure in their system [9] and later upgraded it to RIBL2 in [21]. The calculation procedure is summarized as follows: For two relational data objects $\mathbf{x}_i$ and $\mathbf{x}_j$ of concept $c_s$, the relational similarity measure

$$fs(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{|MC(c_s)|} \sum_{c_t \in MC(c_s)} w_{st} \cdot fs_{set}(\mathbf{x}_i.c_t, \mathbf{x}_j.c_t) \tag{1}$$

where weight $w_{st}$ ($w_{st} \leq 1$ and $\sum_t w_{st} = 1$) represent the importance of member concept $c_t$ when describing the concept $c_s$. In Equation 1, $fs_{set}(\mathbf{x}_i.c_t, \mathbf{x}_j.c_t)$ is defined as:

$$fs_{set}(\mathbf{x}_i.c_t, \mathbf{x}_j.c_t) = \begin{cases} \frac{1}{|\mathbf{x}_i.c_t|} \sum\limits_{\mathbf{y}_l \in \mathbf{x}_j.c_t} \max\limits_{\mathbf{y}_k \in \mathbf{x}_i.c_t} fs(\mathbf{y}_k, \mathbf{y}_l), \\ \qquad \text{if } |\mathbf{x}_i.c_t| \geq |\mathbf{x}_j.c_t| > 0. \\ \frac{1}{|\mathbf{x}_j.c_t|} \sum\limits_{\mathbf{y}_k \in \mathbf{x}_i.c_t} \max\limits_{\mathbf{y}_l \in \mathbf{x}_j.c_t} fs(\mathbf{y}_k, \mathbf{y}_l), \\ \qquad \text{if } |\mathbf{x}_j.c_t| \geq |\mathbf{x}_i.c_t| > 0. \\ 0, \qquad \text{if } |\mathbf{x}_i.c_t| = 0 \text{ or } |\mathbf{x}_j.c_t| = 0. \end{cases} \tag{2}$$

If $MC(c_j) \neq \emptyset$, the value of $fs(\mathbf{y}_k, \mathbf{y}_l)$ in Equation 2 is recursively calculated by Equation 1. It means to replace $\mathbf{x}_i$ (resp. $\mathbf{x}_j$) by $\mathbf{y}_k$ (resp. $\mathbf{y}_l$) in Equation 1 and then use Equation 2 to compare all the associated objects that are referenced by $\mathbf{y}_k$ or $\mathbf{y}_l$. This relational similarity measure explores the linkage structure of the relational objects in a recursive fashion. The procedure continues until $MC(c_j) = \emptyset$ or the depth bound is reached, where the propositional similarity metrics can be applied.

Recursively considering all the associated data objects in Equations 1 and 2 by exhaustively exploring the whole structure of relationships is neither feasible nor necessary in practice. We can quantitatively evaluate the influence of similarity values between data objects in the $d$-th level to the comparison of root data objects. From Equation 1, we see that the similarity value between two member objects of concept $c_t$ will be propagated into the similarity calculation of the upper level concept $c_s$ with a decay factor $\delta_d(c_t) = \frac{w_{st}}{|MC(c_s)|}$, where $d$ means concept $c_t$ is located at the $d$-th level of the root object's relational structure. The total decay factor for concept $c_t$ to impact the similarity calculation of two root objects is $\Delta(c_t) = \prod_d \delta_d$. In many applications, this factor will reduce very quickly as $d$ increases, which means the impact from member objects at the deeper levels of the relational structure decreases [28]. Hence it is not unusual for a depth limit to be set when computing similarity between objects.

### *B. A Taxonomy of Multi-Relational Clustering Approaches*

Multi-relational clustering algorithms can be classified into distance-based, reinforcement, model-based, graph theoretic approaches and user-guided or constraint based clustering. Of these reinforcement based clustering is the only approach that does not have its roots in propositional techniques for clustering.

*1) Distance-based Clustering:* The development of similarity metrics for multi-relational data as introduced in the previous section led to the development of clustering algorithms based on standard propositional clustering algorithms for partitional and hierarchical clustering. Kirten and Wrobel developed RDBC [22]and FORC [23] as the extensions of classic hierarchical agglomerative and k-partitional clustering algorithms respectively. Both of them adopt the distance measure RIBL2 [21] to calculate the dissimilarity between data objects and keep the same algorithmic procedure as their propositional ancestors. Because relational data objects are not additive and divisible as numeric vectors, RDBC and FORC use the *medoid* to represent the clusters in their execution. The medoid of a cluster is defined as the data object that has the maximum average similarity (or minimum average of distance) with the other objects in that cluster. This requires the comparison between every pair of data objects in the given cluster, which has quadratic computational complexity. Therefore, RDBC and FORC are not suitable for clustering very large datasets as they have time-complexities in $O(n^2)$.

*2) Reinforcement Clustering:* The principal of reinforcement clustering approaches comes from the observation that, since all data objects of different types are inter-related to each other, the cluster result of one data type might be propagated along the relationship structure to improve that of other data types. Anthony and desJardins summarized this idea as *inter-cluster relation signature* in [1]: First, data objects of a certain type, say $c_i$, are clustered based on their attributes using some propositional clustering methods. Then for objects of type $c_j$ referencing (or being referenced by) $c_i$, the inter-cluster relation signature is constructed as an $K$-dimensional vector, where $K$ is the number of clusters obtained through the clustering of $c_i$. The value of each dimension in the inter-cluster relation vector $v_k$ is the number of edges that an object of type $c_j$ has if they are linked to objects of type $c_i$ in cluster $k$. The above procedure is iterated for all data types, until the clusters become stable.

The relational clustering algorithm motivated by the idea of mutual reinforcement was firstly implemented by Zeng et al. in [45] in the scenario of clustering heterogeneous web objects, such as web-pages/users or queries/documents. They carefully analyzed several cases of mutual reinforcement in clustering, and concluded that "when the links among nodes are dense enough and contain mostly correct information", the cluster results of one data type will improve that of other related data types. Additionally, they used a hybrid similarity function, a weighted sum of two similarity measures based on content features and link features, to compare data objects during the clustering procedure. The ReCoM framework proposed by Wang et al. [40] further developed this idea by incorporating the importance of data objects to improve the cluster quality. Besides being used to group data objects in an iterative reinforcement fashion, the relationship structure is also used to differentiate the importance of data objects. More important

objects, just as authoritative and hub nodes in the HITS algorithm [25], can have more influence on the clustering procedure. When clustering data objects of a certain type, both of the above frameworks transform the information of relationship structure into the link feature vector of the current objects according to the cluster result of other data types. By this means, the relational clustering task is propositionalized and thus traditional clustering algorithms could be easily embedded into each iterative clustering step to improve the efficiency.

*3) Model-based Clustering:* From the viewpoint of probability theory, data objects are assumed to be generated by a set of parametric probability models. The derived clusters, which are expected to reflect the natural structures hidden in the dataset, should match the underlying models. These probability distributions might be of different types or have the same density function but with different parameters. If the distributions are known, finding clusters within a given data set is equivalent to the problem of estimating parameters for underlying models. The EM (Expectation-Maximization) algorithm is a popular solution for this kind of problem [4]. In propositional datasets, mixtures of multivariate Gaussian distributions are often used due to its well-developed theoretical foundation [11][12].

Taskar et al. [38] propose a general class of models for classification and clustering in relational domains. All relational instances are modeled by the framework of Probabilistic Relational Models (PRMs), in which the attributes of an instance are, based on a conditional probability distribution, determined by the related attributes of its parent instances. The parameters of PRMs are learned from data by utilizing the EM algorithm. Since the networks would be fairly complex, they have to adopt the strategy of belief propagation as the approximation scheme. Liu et al. [30] extends the reinforcement relational clustering framework proposed by Zeng et al. [45]. They introduce two latent clustering layers, serving as two mixture probabilistic models to derive object features. The EM algorithm is used iteratively to estimate the parameters of the mixture models in each layer.

*4) Graph-theoretic Clustering:* Data clustering and graph partitioning, a sub domain in graph theory [42], share many common traits, so we can easily use concepts and techniques of graph partitioning to define and solve the problem of data clustering: A weighted graph $G = \{V, E\}$ will be constructed to describe the target dataset $D$, where vertices $V$ correspond to all data objects in $D$ and the weight of edges $E$ reflect the proximities between each pair of data points. Graph-based approaches can be further classified into Graph Partitioning and Spectral clustering based approaches.

- Graph Partitioning based approaches view clustering of data objects in $D$ as equivalent to finding highly connected sub-graphs in $G$ so that some prescribed properties are optimized, for example minimizing the sum of edge weights on the cut or maximizing the sum of edges weights within the partitioned sub-graphs. Chameleon [20] is a good example that incorporate graph partitioning techniques in propositional cluster analysis.
  In relational clustering, Neville et al. [32] provided some preliminary work of adapting graph-based techniques to incorporate both linkage structure and attribute information. In their paper, the similarity of a pair of related objects is determined by the number of common attributes they share. Objects that are not directly related in the linkage structure have zero similarity regardless of their attribute values. This similarity measure is used to weight edges of the graph $G$. Then three algorithms, Karger's Min-Cut [19], MajorClust [36] and spectral clustering with normalized-cuts criterion [35], are used to partition the graph $G$ and thus generate the clustering result for the original dataset $D$. One disadvantage of this approach is that only the information contained in the first degree of linkage structure is exploited in clustering. It does not consider the pairwise similarity between data objects that are linked via more than one intermediate objects.
- Spectral Clustering has recently developed into one of the most popular approaches to clustering [34][14]. Rooted in the spectral theory [37], spectral clustering approaches need to analyze the eigenvectors of an affinity matrix (also called as "similarity matrix" or "adjacent matrix") derived from the dataset. The problem of optimally partitioning the original dataset is usually converted into solving a set of algebraic equations [35] or performing the traditional clustering procedure in a new feature space spanned by the eigenvectors [33]. Finally it is necessary to interpret the cluster indices for the original dataset from the computational result. Weiss reviewed several typical spectral clustering algorithms within a unified framework [41]. Since spectral clustering can handle non-sphere clusters and is easily to be implemented, it has been successfully applied in the field of speech separation, image segmentation, bio-data classification, etc.
  Long et al. presented a general framework for multi-type relational clustering in [31]. Based on the assumption that the hidden structure of a data matrix can be explored by its factorization, the multi-type relational clustering is converted into an optimization problem: approximate the multiple relation matrices and the feature matrices by their corresponding collective factorization. Under this model a spectral clustering algorithm for multi-type relational data is derived, which updates one intermediate cluster indicator matrix as a number of leading eigenvectors at each iterative step until the result converges. Finally the intermediate matrices have to be post-processed to extract the meaningful cluster structure.

*5) Semi-supervised and User-guided Clustering:* Until now all the relational clustering approaches we have introduced belong to unsupervised learning, which means they are performed under the assumption that there is no preliminary knowledge about the distribution of the dataset. Recently semi-supervised [3][24] and user-guided clustering [44] also attract research interest, where users can provide information or opinion to influence the clustering procedure. These two methodologies are quite different from classification, since the user's knowledge might be inaccurate or incomplete. Hence, it is necessary that the clustering algorithms have the

DIVA (dataset $\mathcal{D}$, number of ROs $r$, variance $v$)

1) cluster set $\{\mathcal{C}_k\} \leftarrow$ call the *Divisive-Step*, given $\mathcal{D}$, $r$ and $v$ as the parameters.
2) dendrogram $T \leftarrow$ call the *Agglomerative-Step*, given $\{\mathcal{C}_k\}$ as the parameter.
3) determine the appropriate level in $T$ to construct the clustering result.

TABLE I
MAIN FRAMEWORK OF DIVA

capabilities of automatically evaluating the pertinence of the given information and utilizing them appropriately.

## III. A NOVEL AND EFFICIENT RELATIONAL CLUSTERING FRAMEWORK - *DIVA*

In this section, we will introduce a general clustering framework *DIVA* for relational datasets [28]. DIVA belongs to the class of distance-based approaches to clustering. The objective of DIVA is three-fold:

- To introduce an approach to distance-based relational clustering that scales well with the number of objects being clustered. Currently distance-based approaches such as FORC and RDBC have quadratic time complexity and are hence unsuitable for use within large data sets.
- To be robust to outliers.
- To provide a flexible framework to support the discovery of arbitrary shaped clusters.

These objectives are met by DIVA through the use of three key concepts. These are the use of multiple representation objects to represent each cluster, the use of a multi-phase approach to clustering consisting of a divisive step followed by an agglomerative step and the use of a variance threshold during the divisive step. We now describe each of these concepts and explain how they deliver the objectives set out above.

Like propositional clustering algorithm BIRCH [46] and Chameleon [20], DIVA uses a multi-phase approach to clustering: divisive and agglomerative. The whole dataset is first divided into a number of clusters so that the variance of each cluster is equal to or less than a particular threshold value $v$. Based on these clusters, a hierarchical dendrogram is built using an agglomerative approach. An appropriate level of the dendrogram is determined to construct the final cluster result. The whole procedure of the clustering framework DIVA is summarized in Table I.

Section III-A introduces the idea of Representative Objects and its application in the procedure of clustering. Based on that, Section III-B describes the divisive (including recursive and incremental approaches) as well as the agglomerative steps of DIVA. Section III-C briefly analyze the computational complexity of each step in DIVA. And finally some experimental results are provided in Section III-D.

### A. Representative Objects

Distance-based partitional clustering algorithms have traditionally used a single prototypical object from each cluster (typically the centroid or medoid) to allocate objects to the clusters and to compute the quality of the resulting clustering. The sensitivity of the centroid to outliers makes the medoid a more robust prototype, however, medoid based algorithms such as PAM are known to be more computationally expensive. Furthermore, such approaches lend themselves to the discovery of spherical clusters of similar sizes. CURE [13] proposed the use of multiple prototypical objects (called Representative Points, RP) and used an agglomerative approach to hierarchically clustering objects in propositional datasets. The advantage of using multiple prototypical objects was the ability to discovery arbitrarily shaped clusters. The RPs were defined using the concept of maximum spread though the chosen RPs were "shrunk" by a factor, $\alpha$, towards the centroid to provide robustness to outliers.

DIVA also uses multiple maximum spread objects, called Representative Objects (ROs), to represent clusters. However, in the absence of a centroid in multi-relational space and to avoid the computational cost of computing the medoid, DIVA uses a variance threshold to ensure robustness to outliers. In addition to the advantage that multiple ROs provide with respect to the discovery of arbitrary shaped clusters, as discussed in Section III-C, ROs provide the basis for reducing the time complexity of multi-relational clustering algorithms.

As stated previously, ROs are defined as a set of maximum-spread objects in the dataset $\mathcal{D}$, denoted as $\{ro_i\}$ $(1 \leq i \leq r)$. Moreover, the distance between the farthest pair of ROs approximates to the diameter of the data space. We define the *variance* of the dataset $\mathcal{D}$ as:

$$Var(\mathcal{D}) = \max_{1 \leq i,j \leq r} fd_{obj}(ro_i, ro_j) \tag{3}$$

Small variance means data objects reside in a compact data space and thus are more similar to each other.

An efficient method for determining the ROs was developed in [28]. After a start object $x_s$ is randomly chosen, the $i$-th RO is determined as follows:

$$ro_i = \begin{cases} \arg\max_{x \in \mathcal{D}} fd_{obj}(x, x_s) & \text{if } i = 1 \\ \arg\max_{x \in \mathcal{D}} \left( \min_{1 \leq j < i} fd_{obj}(x, ro_j) \right) & \text{if } 2 \leq i \leq r \end{cases} \tag{4}$$

where $fd_{obj}(\cdot, \cdot)$ is the distance measure for relational data objects. In Section III-C we will show that the computational complexity of the above method is linear to the size of the dataset $\mathcal{D}$. However, Equation 4 is not suitable in the scenario of incremental learning: every time a new data object is added into the dataset, re-selecting all the ROs from scratch with Equation. 4 will lead to quadratic complexity.

An alternative method for dynamically determining ROs, which is especially suitable for the scenario of incremental learning, is developed in [29]. Consider a collection of data objects $\mathcal{D}$, of which the set of ROs $\{ro_i\}$ $(1 \leq i \leq r)$ have been selected. Assume a new data object $x$ is added to $\mathcal{D}$, one of the existing ROs would be replaced by $x$ if the new RO set holds the maximum-spreading requirement. Theoretically there are $r + 1$ possible combinations to be examined within the set $\{ro_i\} \cup \{x\}$. If all the pairwise similarity values between existing ROs have been stored in the memory, both

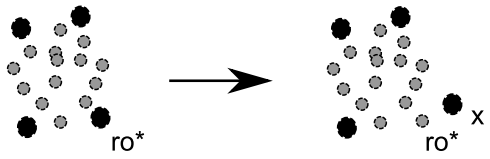


Fig. 3. Dynamically update the ROs

Recursively-Divisive-Step(dataset $\mathcal{D}$, number of ROs $r$, variance $v$)

1) CREATE the initial cluster $\mathcal{C}_0$ containing all the data in $\mathcal{D}$. Insert $\mathcal{C}_0$ into the list $L$.
2) FOR EACH newly inserted cluster $\mathcal{C}_k$ in $L$:
   a) generate the set of ROs $\{ro_i^{(\mathcal{C}_k)}\}$:
      i) the start object $x_s^{(\mathcal{C}_k)} \leftarrow$ randomly select an object from $\mathcal{C}_k$.
      ii) to determine $ro_i^{(\mathcal{C}_k)}$ $(1 \le i \le r)$:
      $ro_i^{(\mathcal{C}_k)} \leftarrow$ select the object $x$ from $\mathcal{C}_k$ that is farthest away from the start point $x_s^{(\mathcal{C}_k)}$ when $i = 1$ or that maximize the accumulated distances from itself to the already obtained ROs $ro_j^{(\mathcal{C}_k)}$ $(1 \le j < i)$ when $2 \le i \le r$, as described in Equation 4.
   b) evaluate the variance of $\mathcal{C}_k$ by Equation 3. Without loss of generality, assume the pair of ROs in $\{ro_i^{(\mathcal{C}_k)}\}$ that are farthest away from each other are $ro_1^{(\mathcal{C}_k)}$ and $ro_2^{(\mathcal{C}_k)}$. Then $Var(\mathcal{C}_k) = fd(ro_1^{(\mathcal{C}_k)}, ro_2^{(\mathcal{C}_k)})$.
   c) if $V(\mathcal{C}_k) > v$, then:
      i) create two new clusters $\mathcal{C}_{k'}$ and $\mathcal{C}_{k''}$, using $ro_1^{(\mathcal{C}_k)}$ and $ro_2^{(\mathcal{C}_k)}$ as the absorbent objects of $\mathcal{C}_{k'}$ and $\mathcal{C}_{k''}$ respectively, where $k'$ and $k''$ are unused index numbers in $L$.
      ii) allocate the rest objects $x \in \mathcal{C}_k$ into either $\mathcal{C}_{k'}$ or $\mathcal{C}_{k''}$ based on the comparison of $fd(x, ro_1^{(\mathcal{C}_k)})$ and $fd(x, ro_2^{(\mathcal{C}_k)})$.
      iii) insert $\mathcal{C}_{k'}$ and $\mathcal{C}_{k''}$ into $L$ to replace $\mathcal{C}_k$.
3) RETURN all the remaining clusters $\mathcal{C}_k$ in $L$.

TABLE II
THE RECURSIVE DIVISIVE STEP

the time and the space complexities of this method are $O(r^2)$. To further improve the efficiency, we incorporate a heuristic method to update the RO set: for the new data $x$, we can find its nearest RO in $\{ro_i\}$, denoted as $ro^*$. The replacement of $ro^*$ by $x$ happens only if the following condition is true:

$$\min_{\substack{1\le i\le r,\\ ro_i \ne ro^*}} fd_{obj}(x, ro_i) > \min_{\substack{1\le j\le r,\\ ro_j \ne ro^*}} fd_{obj}(ro^*, ro_j) \qquad (5)$$

To perform comparison in Equation 5, we only need to store the minimum distance of each RO to the other ROs, so both the space and the time complexities are reduced to $O(r)$. Figure 3 shows the above procedure: when a new data object $x$ is added, the original RO $ro^*$ is replaced because $x$ can better represent the shape of the whole dataset.

### *B. Divisive and Agglomerative Steps*

Corresponding to the methods of determining ROs discussed in the previous section, there are two approaches to dividing the original dataset $\mathcal{D}$: recursive (based on Equation 4) and incremental (based on Equation 5).

Incremental-Divisive-Step(dataset $\mathcal{D}$, number of ROs $r$, variance $v$)

1) Randomly select a data object $x_0 \in \mathcal{D}$ to form the start cluster $\mathcal{C}_0$. Insert $\mathcal{C}_0$ into the cluster list $L$.
2) FOREACH data object $x \in \mathcal{D}$:
   a) Select the nearest cluster $\mathcal{C}^*$ to $x$ using Equation 7. Assume $\{ro_i^{(\mathcal{C}^*)}\}$ is the set of ROs for $\mathcal{C}^*$, among which $ro^*$ is the nearest one to $x$.
   b) IF $\max_{1\le i\le r} fd_{obj}(x, ro_i^{(\mathcal{C}^*)}) \le v$:
      i) Add $x$ into the cluster $\mathcal{C}^*$.
      ii) IF the size of $\{ro_i^{(\mathcal{C}^*)}\}$ is less than $r$, $x$ will be added into $\{ro_i^{(\mathcal{C}^*)}\}$ as a new RO.
      ELSE $ro^*$ will be replaced by $x$ in $\{ro_i^{(\mathcal{C}^*)}\}$ when the Inequality 5 is true.
   ELSE Create a new cluster $C'$ which only contains $x$. Then insert $C'$ into $L$.
3) RETURN the cluster list $L$.

TABLE III
THE INCREMENTAL DIVISIVE STEP

The recursive divisive procedure is described in Table II. Given the whole dataset $\mathcal{D}$ as the initial cluster $\mathcal{C}_0$, its ROs are selected by Equation 4 and the variance of $\mathcal{C}_0$ is calculated by Equation 3. If $Var(\mathcal{C}_0)$ is greater than the pre-specified variance threshold $v$, $\mathcal{C}_0$ will be divided. Two furthest ROs are used as the absorbent objects of sub-clusters respectively, and the other data objects in $\mathcal{C}_0$ are distributed according to their distance to the absorbent objects. Then for each of derived sub-clusters, the above procedure is recursively launched when its variance does not satisfy the variance requirement.

The incremental divisive step follows the idea of traditional first leader clustering algorithm [17]: each new data point is added into its nearest cluster, of which the center to the current point is less than an Euclidean distance *MaxDistance*, and the cluster's center is updated accordingly; if no such cluster exists, the point will form a new cluster by itself. In our clustering procedure, all the data objects in $\mathcal{D}$ are scanned sequentially. The MaxDistance constraint for relational data is defined as:

$$\max_{1\le i\le r} fd_{obj}(x, ro_i^{(\mathcal{C})}) \le v \qquad (6)$$

where $\{ro_i^{(\mathcal{C})}\}$ is the set of ROs for cluster $\mathcal{C}$ and $v$ is the variance that controls the compactness of all derived clusters. The above constraint guarantees that the newly generated cluster $\mathcal{C}'$ still holds the requirement of variance and the previous absorbed data objects are still valid, avoiding the unbound problem in the original First Leader method. In the case that more than one candidate clusters is available, we adopt the complete-linkage strategy to pick out the appropriate cluster $\mathcal{C}^*$ to absorb $x$:

$$\text{Complete-linkage: } \mathcal{C}^* = \arg\min_{\mathcal{C}} \left( \max_i fd_{obj}(x, ro_i^{(\mathcal{C})}) \right) \qquad (7)$$

The single-linkage and average-linkage strategies may also be adopted according to the requirement of different applications:

$$\text{Single-linkage: } \mathcal{C}^* = \arg\min_{\mathcal{C}} \left( \min_i fd_{obj}(x, ro_i^{(\mathcal{C})}) \right)$$

Agglomerative-Step(cluster set $\{\mathcal{C}_k\}$)

1) INITIALIZE the dendrogram $T$. For each $\mathcal{C}_k$, construct a leaf node $t_k$ in $T$.
2) REPEAT for $K-1$ times, where $K$ is the size of $\{\mathcal{C}_k\}$:
   a) for nodes in $T$ that have no parent node, their pairwise similarity values are evaluated by Equation 8. From all these nodes, we choose the pair with the highest similarity value, assuming they are $t_l$ and $t_{l'}$.
   b) generate a new node $t_p$ as the parent node for both $t_l$ and $t_{l'}$, which equals to create a new super-cluster $\mathcal{C}_p$ by merging $\mathcal{C}_l$ and $\mathcal{C}_{l'}$. The top-$r$ maximum-spread ROs in $\{ro_i^{(t_l)}\} \cup \{ro_j^{(t_{l'})}\}$ are chosen as the ROs for $t_p$.
   c) store $t_p$ into $T$.
3) RETURN $T$.

TABLE IV
THE AGGLOMERATIVE STEP

$$\text{Average-linkage: } \mathcal{C}^* = \arg\min_{\mathcal{C}} \left( \frac{1}{r} \sum_i fd_{obj}(x, ro_i^{(\mathcal{C})}) \right)$$

The incremental divisive step is summarized in Table III.

When the divisive step is finished, we get a set of clusters $\{\mathcal{C}_k\}$ ($\bigcup_k \mathcal{C}_k = \mathcal{D},\ \mathcal{C}_{k_1} \bigcap \mathcal{C}_{k_2} = \emptyset$) with variance equals or is less than $v$. Note that the use of the variance threshold as a parameter in defining these clusters implies a bias towards spherical clusters.

The clusters obtained at the end of the divisive step are used as input to the agglomerative step. They constitute the leaf nodes of the dendrogram resulting from the agglomerative step in DIVA. In each iteration, the most similar pairwise sub-clusters (child-nodes) are merged to form a new super-cluster (parent-node)[1]. The similarity between two nodes is calculated using their set of ROs:

$$fd_{node}(t_l, t_{l'}) = \max_{i,j} fd_{obj}(ro_i^{(t_l)}, ro_j^{(t_{l'})}) \tag{8}$$

where $\{ro_i^{(t_l)}\}$ and $\{ro_j^{(t_{l'})}\}$ are the sets of ROs contained in node $t_l$ and $t_{l'}$ respectively, assuming the super node $t_p$ is formed from two sub-nodes $t_l$ and $t_{l'}$. Note that this approach to merging clusters in equivalent to *complete-linkage* using only the RO sets of the clusters. The agglomerative step is summarized in Table IV. It is worth noting here that the agglomerative step is not a reverse reproduction of the recursive divisive step. As shown in BIRCH [46] and Chameleon [20], the agglomeration can remedy the inaccurate partitioning generated by the divisive step. Note that the use of *single-linkage* as opposed to Complete-linkage provides the flexibility to discover arbitrarily shaped clusters using DIVA.

After the dendrogram $T$ is built, an appropriate level in $T$ is determined to construct the final cluster result. A common strategy is to select the level at which the variance of each node equals or is less than $v$. Alternatively, we can record the variance of newly generated nodes for each level, find the largest gap between variances of two neighbored levels and use the lower level as the basis to construct clusters [6]. When the number of clusters is fixed, the level which contains the exactly required number of nodes is selected to construct clusters.

[1]Since each cluster in the agglomerative step is related to a unique node in the dendrogram, the words "cluster" and "node" are used interchangeably here.

### C. Complexity Analysis

In this section, we briefly analyze the computational complexity for each step in DIVA, given the whole dataset $\mathcal{D}$ of size $N$, the number of maximum iteration in the divisive step is $R$ and the size of $\{\mathcal{C}_k\}$ is $K$.

- For the recursive divisive step based on Equation 4:
  - After the start object $x_s$ is randomly selected with complexity $O(1)$, the first RO $ro_1$ is determined by scanning the initial dataset $\mathcal{C}_0$ once to pick out the farthest data object from $x_s$. Similarly, the determination of each $ro_i$ $(2 \leq i \leq r)$ only needs to scan the whole dataset once by comparing all the non-RO objects with $ro_{i-1}$, because the other required similarity values have been obtained when determining $ro_j (1 \leq j \leq i-2)$. Overall, the computational complexity is $O(r \cdot N)$.
  - When the dataset $\mathcal{C}_k$ has to be divided, two sub-clusters are constructed by appointing the farthest pair of ROs (assuming $ro_i$ and $ro_j$) as the absorbent objects respectively. Then, all the non-RO objects can be allocated to the nearest cluster without extra calculation because their similarity values to $ro_i$ or $ro_j$ have been obtained in the procedure of determining $\{ro_i\}$. Hence, the operation for dividing and redistributing dataset $\mathcal{D}$ has complexity $O(1)$. The above division procedure is launched recursively for all the derived sub-clusters until their variances satisfy the requirement. Assume the cluster $\mathcal{C}_k$ of size $N_k$ in the final cluster result is generated from the original dataset $\mathcal{D}$ by at most $R$ iterations, then all the data objects in $\mathcal{C}_k$ need to be compared with $R \cdot r$ ROs during the recursive division. Hence, the total computational complexity for the recursive division is $O(\sum_k R\, r N_k)$, i.e. $O(R\, r N)$.
- For the incremental divisive step based on Equation 5:
  - when a new data object is added, it will compared with all the ROs in previous created clusters. The computational complexity in the worse case is $O(K \cdot r)$.
  - Since the whole dataset $\mathcal{D}$ will be scanned only once, the total computational complexity is hence $O(K\, r N)$.
- For the agglomerative step based on Equation 8:
  - every pair of ROs in two different clusters will be compared, so the computational complexity of building the taxonomy is $O(r^2 \cdot K^2)$.

In summary, the total computational complexity of the DIVA algorithm is $O(rN + rRN + r^2K^2)$ when using the recursive divisive fashion or $O(N \cdot r \cdot K + r^2K^2)$ when using the incremental divisive fashion. In the case that $rR \ll N$ or $rK \ll N$, the computational complexity of our DIVA algorithm would be linear to the size of the dataset. When the

whole dataset can be stored in the memory and the data objects are evenly distributed, the recursive method is usually better, because we have $K = log_2 R$ when the derived divisive tree is balanced. In contrast, the incremental division is superior for processing very large datasets because the data will be scanned only once, while recursive division might access each data object many times resulting expensive disk I/O operations.

It is worth noting that $R$ in the recursive division is controlled by the variance threshold $v$: lower $v$ leads to more recursive operations and thus generates more clusters. In the incremental division $K$ is controlled by $v$ in the same way. When $v \to 0$, too many tiny clusters (or singular clusters in the extreme) will be generated, which makes the agglomerative step behave like the pure agglomerative approach RDBC with quadratic complexity. Hence, the value of variance $v$ should be set appropriately to avoid such unnecessary division. A good strategy is to gradually increase the value of $v$ to improve the homogeneity of the generated clusters, until their quality meets our requirement.

### D. Experimental Results

Some comprehensive experiments are conducted to examine the efficiency and effectiveness of our algorithm. For the sake of brevity, in this section we use the word "DIVA" to stand for the implementation of DIVA consisting of the recursive division plus the agglomeration, while the word "HIREL" means the incremental division plus the agglomeration. They are compared with the following well-known multi-type relational clustering algorithms: (1) ReCoM [40], which uses relationships among data objects to improve the cluster quality of interrelated data objects through an iterative reinforcement clustering process. Because there is no prior knowledge about the authoritativeness in the datasets, we treat all data objects as equally important. Additionally, k-Medoids is incorporated as the meta clustering approach in ReCoM. (2) FORC [23], which is a natural extension of k-Medoids in the field of relational clustering.

To evaluate the accuracy of the clustering result, the criterion of Related Minimum Variance [6] is adopt to measure the similarity between pairs of objects in the same cluster.

$$S_{intra} = \frac{1}{K} \sum_{k} \Big( \frac{1}{N_k^2} \sum_{x \in \mathcal{C}_k} \sum_{x' \in \mathcal{C}_k} fs_{obj}(x, x') \Big)$$

where $N_k$ is the size of cluster $\mathcal{C}_k$. Generally speaking, higher intra-cluster similarity indicates higher quality of derived clusters. Another criterion is based on entropy to evaluate the uniformity or purity of a cluster, if the class labels of data objects are available [40].

$$H(\mathcal{C}_k) = -\sum_{h} P_{h,k} \log_2 P_{h,k} \tag{9}$$

where $P_{h,k}$ is the proportion of data objects of class $h$ in the cluster $\mathcal{C}_k$. The total entropy of the cluster result is the sum of entropies across all clusters. Different from the intra-cluster similarity measure, lower entropy value means more homogenous clusters.

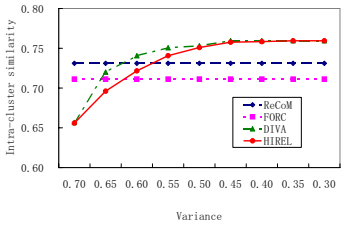


Fig. 4. Intra-cluster similarity (Synthetic Amazon Data)

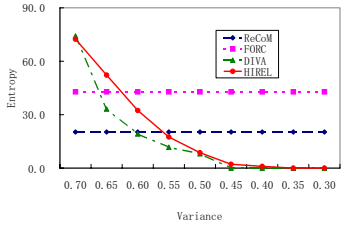


Fig. 5. Entropy (Synthetic Amazon Data)

*1) Synthetic Amazon Dataset:* This dataset simulates the users' browsing products on the website *http://www.amazon.com* based on their interests. The process of data generation has been explained in [28]. In total there are 2000 users, 10,000 virtual products and 100,000 browsing actions. Here we keep the parameters for algorithms ReCoM, FORC and DIVA the same as in [28]. Each clustering algorithm generates 100 final clusters for users.

Figures 4 - 6 show the evaluation results, changing $v$ from 0.7 to 0.3 and fixing $r = 3$ for DIVA and HIREL. As we expect, the quality of clusters derived by DIVA and HIREL will be greatly improved as $v$ decreases. When $v < 0.55$, both DIVA and HIREL outperform the other two algorithms. DIVA converges more quickly than HIREL, but their quality of the clustering result are almost the same when $v$ is sufficient small ($v < 0.5$ in this case). From Figure 6 we can see that the time spent by HIREL is far less than that of the other three algorithms.

Figures 7 and 8 illustrate the robustness of all approaches under different noise ratios of browsing actions, ranging from 20% to 100%. We fixed $v = 0.5$ and $r = 3$ for DIVA and

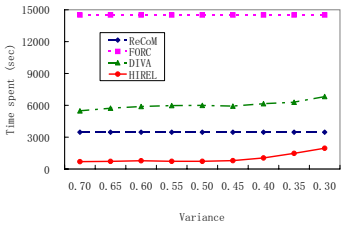


Fig. 6. Time spent (sec) (Synthetic Amazon Data)

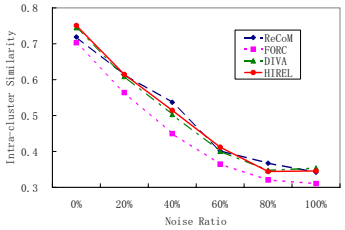


Fig. 7. Intra-cluster similarity (Synthetic Amazon Data with Noise)

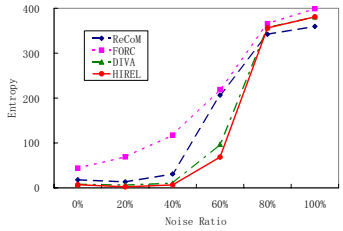


Fig. 8. Entropy (Synthetic Amazon Data with Noise)

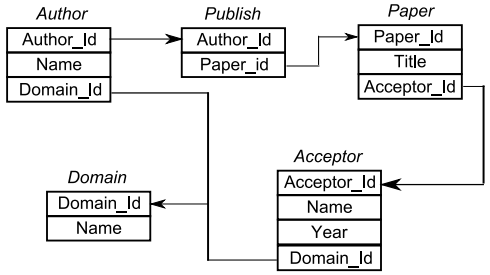


Fig. 9. Schema of the DBLP database

HIREL. In general, the accuracy of all approaches are reduced as noise ratio increases. Being evaluated by the entropy-based criterion, DIVA and HIREL exceed the other two algorithms when the noise ratio is below 80% and performances of all the four algorithms are very close when the noise ratio is above 80%, which means DIVA and HIREL are very suitable for clustering datasets with reasonable noise ratio.

*2) DBLP Dataset:* This dataset is download from the website *http://dblp.uni-trier.de/*, containing thousands of authors, papers, conferences as well as their relationships. Figure 9 shows the database schema. we selected 503 authors and 615 acceptors in our analysis. In total there are 83579 papers written by these authors and accepted by the acceptors. In order to evaluate the cluster result, we manually labeled the research areas of all the authors by assigning them into (up to three) domains. Then the Kullback-Leibler Divergence [6] is used to evaluate the relative entropy of the clusters, by changing Equation 9 as:

$$H(\mathcal{C}_k) = \sum_h P_{h,k} \log_2 \frac{P_{h,k}}{Q_{h,k}}$$

where $Q_{h,k}$ is the expected proportion of data objects of class $h$ in the cluster $\mathcal{C}_k$.

Figures 10 - 12 report our experimental results. Since the linkage structure of this dataset is much denser than that of the Amazon dataset, ReCoM generates the worst cluster result ($divergence = 2.0$) and spent the longest time

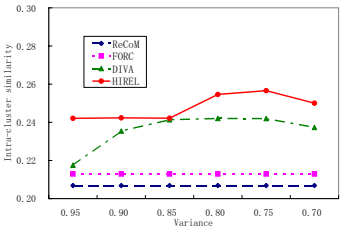


Fig. 10. Intra-cluster similarity (DBLP Data)

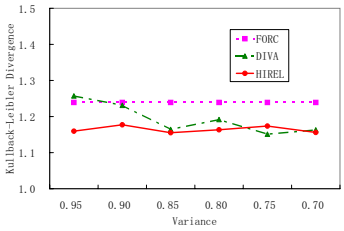


Fig. 11. KL Divergence (DBLP Data)

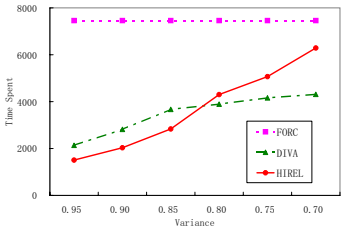


Fig. 12. Time spent (sec) (DBLP Data)

($TimeSpent > 100Ksec$), so it is excluded in Figure 11 and 12. Figure 10 and Figure 11 show that the cluster results derived by DIVA and HIREL are comparatively homogenous, which are much better than those of ReCoM and FORC.

Generally as $v$ decreases, the time spent by DIVA and HIREL are increased accordingly, but they are still faster than ReCoM and FORC. When $v \leq 0.8$, the time spent by HIREL is more than that of DIVA. This phenomenon is understandable because the incremental divisive step of HIREL generated more than 250 clusters in such case. Considering the total number of authors is 503, each derived cluster only contained two authors in average. Such unnecessary division makes the HIREL behave like the pure agglomerative algorithm with quadratic complexity, as we have discussed before. Instead, DIVA performs better keep the linear computational complexity due to its recursive division, because, as the recursive division goes deeper, each object will be compared with less and less ROs.

## IV. Conclusions and Future Work

In this paper we described the current state-of-the-art in multi-relational clustering. A taxonomy of approaches to clustering complex multi-relational objects was presented along with representative algorithms proposed in each category. We then described our framework for distance-based clustering, DIVA developed with the aim of developing efficient, effective and flexible multi-relational clustering algorithms. DIVA uses a multi-phase clustering approach consisting of an initial divisive step followed by an agglomerative step. To enhance efficiency of the algorithm, a cluster is represented using a set of maximal spread objects, called Representative Objects. Two implementations of the divisive step, using a recursive and an incremental approach were presented. Empirical evaluation of DIVA shows that DIVA can provide substantial benefits in terms of efficiency while discover better quality clustering than a number of standard algorithms for multi-relational clustering. DIVA also provides benefits with respect to robustness to outliers and the discovery of arbitrary shaped clusters.

In the future, we intend to investigate methods to determine optimal ROs during recursive and incremental clustering.

We will also investigate the application of different cluster similarity measures within the divisive and agglomerative steps within different application scenarios. The experimental results obtained on the DBLP data set warrants further investigation into the use of better data structures for representing clusters when large number of clusters are discovered during the divisive step. Finally, we intend to explore the use of sampling to further improve the time complexity of the algorithm.

## REFERENCES


[1] A. Anthony and M. desJardins. Open problems in relational data clustering. In *Proceedings of ICML Workshop on Open Problems in Statistical Relational Learning*, Pittsburgh, PA, 2006.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[3] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning (ICML'04)*, page 11, New York, NY, USA, 2004. ACM.

[4] J. A. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of California, Berkeley, 1997.

[5] L. Dehaspe and H. Toivonen. Discovery of relational association rules. In *[8]*, pages 189–212. 2001.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience Publication, 2001.

[7] S. Džeroski. Multi-relational data mining: An introduction. *SIGKDD Explorations Newsletter*, 5(1):1–16, 2003.

[8] S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer, September 2001.

[9] W. Emde and D. Wettschereck. Relational instance based learning. In L. Saitta, editor, *Proceedings of 13th International Conference on Machine Learning (ICML)*, pages 122–130. Morgan Kaufmann Publishers, 1996.

[10] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[11] M. Figueiredo and A. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[12] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. Technical Report 380, University of Washington, Dept. of Stats., Oct. 2000.

[13] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, June 1998.

[14] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques (2nd Edition)*. Morgan Kaufmann, 2006.

[16] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Math. Program.*, 79(1-3):191–215, 1997.

[17] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[19] D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms (SODA'93)*, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[20] G. Karypis, E.-H. S. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[21] M. Kirsten, S. Wrabel, and T. Horváth. Distance based approaches to relational learning and clustering. In *[8]*, pages 213–232. 2001.

[22] M. Kirsten and S. Wrobel. Relational distance-based clustering. In *Proceedings of Fachgruppentreffen Maschinelles Lernen (FGML-98)*, pages 119–124, 10587 Berlin, 1998. Techn. Univ. Berlin, Technischer Bericht 98/11.

[23] M. Kirsten and S. Wrobel. Extending k-means clustering to first-order representations. In *Proceedings of the 10th International Conference on Inductive Logic Programming (ILP'00)*, pages 112–129, London, UK, 2000. Springer-Verlag.

[24] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the nineteenth International Conference on Machine Learning (ICML'02)*, 2002.

[25] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[26] S. Kramer. Structural regression trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 812–819, Cambridge/Menlo Park, 1996. AAAI Press/MIT Press.

[27] S. Kramer, N. Lavrač, and P. Flach. Propositionalization approaches to relational data mining. In *[8]*, pages 262–286. 2001.

[28] T. Li and S. S. Anand. DIVA: A variance-based clustering approach for multi-type relational data. In *Proceedings of the ACM Sixteenth Conference on Information and Knowledge Management (CIKM'07)*, Lisboa, Portugal, 2007.

[29] T. Li and S. S. Anand. Hierarchical leader clustering: A fast clustering algorithm for relational datasets. 2008.

[30] G. Liu, W. Zhu, and Y. Yu. A unified probabilistic framework for clustering correlated heterogeneous web objects. In *Proceeding of the seventh Asia-Pacific Web Conference (APWeb2005)*, Shanghai, China, March 2005.

[31] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd international conference on Machine learning (ICML'06)*, pages 585–592, New York, NY, USA, 2006. ACM Press.

[32] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, 18th International Joint Conference on Artificial Intelligence*, 2003.

[33] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th Advances in Neural Information Processing Systems (NIPS'01)*, 2001.

[34] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[35] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[36] B. Stein and O. Niggemann. On the nature of structure and its identification. In *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'99)*, pages 122–134, London, UK, 1999. Springer-Verlag.

[37] P. Stoica and R. L. Moses. *Introduction to spectral analysis*. Upper Saddle River, N.J. : Prentice Hall, 1997.

[38] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In B. Nebel, editor, *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 870–878, Seattle, US, 2001.

[39] W. Van Laer and L. De Raedt. How to upgrade propositional learners to first order logic: A case study. In *[8]*, pages 235–261. 2001.

[40] J. Wang, H. Zeng, Z. Chen, H. Lu, T. Li, and W.-Y. Ma. ReCoM: Reinformcement clustering of multi-type interrelated data objects. In *Proceedings of the 26th ACM SIGIR conference on Research and development in informaion retrieval (SIGIR'03)*, pages 274–281, New York, NY, USA, 2003. ACM Press.

[41] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the 2nd International Conference on Computer Vision (ICCV)*, pages 975–982, 1999.

[42] D. B. West. *Introduction to graph theory (2nd Edition)*. Upper Saddle River, N.J. : Prentice Hall, 2001.

[43] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transaction on Neural Networks*, 16:645–678, May 2005.

[44] X. Yin, J. Han, and P. S. Yu. CrossClus: user-guided multi-relational clustering. *Data Mining and Knowledge Discovery*, 15(3):321–348, 2007.

[45] H.-J. Zeng, Z. Chen, and W.-Y. Ma. A unified framework for clustering heterogeneous web objects. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'02)*, pages 161–172, Washington, DC, USA, 2002. IEEE Computer Society.

[46] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, 1996.