DB-CSC: A Density-Based Approach for Subspace Clustering in Graphs with Feature Vectors

Stephan Günnemann, Brigitte Boden, and Thomas Seidl

RWTH Aachen University, Germany
{guennemann,boden,seidl}@cs.rwth-aachen.de

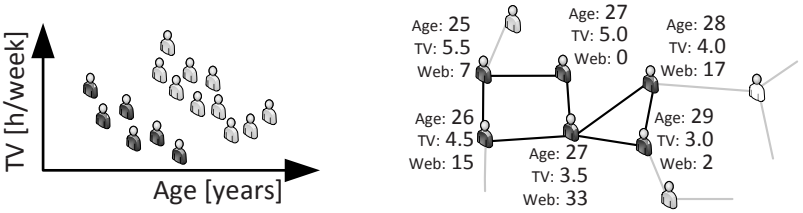
Abstract. Data sources representing attribute information in combination with network information are widely available in today's applications. To realize the full potential for knowledge extraction, mining techniques like clustering should consider both information types simultaneously. Recent clustering approaches combine *subspace clustering* with *dense subgraph mining* to identify groups of objects that are similar in subsets of their attributes as well as densely connected within the network. While those approaches successfully circumvent the problem of full-space clustering, their limited cluster definitions are restricted to clusters of certain shapes.

In this work, we introduce a density-based cluster definition taking the attribute similarity in subspaces and the graph density into account. This novel cluster model enables us to detect clusters of arbitrary shape and size. We avoid redundancy in the result by selecting only the most interesting non-redundant clusters. Based on this model, we introduce the clustering algorithm DB-CSC. In thorough experiments we demonstrate the strength of DB-CSC in comparison to related approaches.

1 Introduction

In the past few years, data sources representing attribute information in combination with network information have become more numerous. Such data describes single objects via attribute vectors and also relationships between different objects via edges. Examples include social networks, where friendship relationships are available along with the users' individual interests (cf. Fig 1); systems biology, where interacting genes and their specific expression levels are recorded; and sensor networks, where connections between the sensors as well as individual measurements are given. There is a need to extract knowledge from such complex data sources, e.g. for finding groups of homogeneous objects, i.e. clusters. Throughout the past decades, a multitude of clustering techniques were introduced that either solely consider attribute information or network information. However, simply applying one of these techniques misses the potential given by such combined data sources. To detect more informative patterns it is preferable to simultaneously consider relationships together with attribute information. In this work, we focus on the mining task of clustering to extract meaningful groups from such complex data.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 565–580, 2011.



(a) Attribute view (2d subspace) (b) Graph view (extract)

Fig. 1. Clusters in a social network with attributes age, tv consume and web consume

Former techniques considering both data types aim at a combination of dense subgraph mining (regarding relationships) and *traditional* clustering (regarding attributes). The detected clusters are groups of objects showing high graph connectivity as well as similarity w.r.t. all of their attributes. However, using traditional, i.e. full-space, clustering approaches on these complex data does mostly not lead to meaningful patterns. Usually, for each object a multitude of different characteristics is recorded; though not all of these characteristics are relevant for each cluster and thus clusters are located only in subspaces of the attributes. In, e.g., social networks it is very unlikely that people are similar within all of their characteristics. In Fig. 1(b) the customers show similarity in two of their three attributes. In these scenarios, applying full-space clustering is futile or leads to very questionable clustering results since irrelevant dimensions strongly obfuscate the clusters and distances are not discriminable anymore [4]. Consequentially, recent approaches [16,9] combine the paradigms of *dense subgraph mining* and *subspace clustering*, i.e. clusters are identified in locally relevant subspace projections of the attribute data.

Joining these two paradigms leads to clusters useful for many applications: In social networks, closely related friends with similar interests in some product relevant attributes are useful for target marketing. In systems biology, functional modules with partly similar expression levels can be used for novel drug design. In sensor networks, long distance reports of connected sensors sharing some similar measurements can be accumulated and transfered by just one representative to reduce energy consumption. Overall, by using subspace clustering the problems of full-space similarity are, in principle, circumvented for these complex data.

Though, the cluster models of the existing approaches [16,9] have a severe limitation: they are limited to clusters of certain shapes. This holds for the properties a cluster has to fulfill w.r.t the network information as well as w.r.t. the attribute information. While for the graph structure models like quasi-cliques are used, for the vector data the objects' attribute values are just allowed to deviate within an interval of fixed width the user has to specify. Simply stated, the clustered objects have to be located in a rectangular hypercube of given width. For real world data, such cluster definitions are usually too restrictive since clusters can exhibit more complex shapes. Considering for example the 2d attribute subspace of the objects in Fig. 1(a): The two clusters can not be correctly detected by the rectangular hypercube model of [16,9]. Either both clusters are merged or some

objects are lost. In Fig. 1(b) an extract of the corresponding network structure is shown. Since the restrictive quasi-clique property would only assign a low density to this cluster, the cluster would probably be split by [9].

In this work, we combine dense subgraph mining with subspace clustering based on a more sophisticated cluster definition; thus solving the drawbacks of previous approaches. Established for other data types, density-based notions of clusters have shown their strength in many cases. Thus, we introduce a *density-based clustering principle* for the considered combined data sources. Our clusters correspond to dense regions in the attribute space as well as in the graph. Based on local neighborhoods taking the attribute similarity in subspaces as well as the graph information into account, we model the density of single objects. By merging all objects located in the same dense region, the overall clusters are obtained. Thus, our model is able to detect the clusters in Fig. 1 correctly.

Besides the sound definition of clusters based on density values, we achieve a further advantage. In contrast to previous approaches, the clusters in our model are not limited in their size or shape but can show arbitrary shapes. For example, the diameter of the clusters detected in [9] is a priori restricted by a parameter-dependent value [17] leading to a bias towards clusters of small size and little extent. Such a bias is avoided in our model, where the sizes and shapes of clusters are automatically detected. Overall, our contributions are:

- We introduce a novel density-based cluster definition taking attributes in subspaces and graph information into account.
- We ensure an unbiased cluster detection since our clusters can have arbitrary shape and size.
- We develop the algorithm DB-CSC to determine such a clustering solution.

2 Related Work

Different clustering methods were proposed in the literature. Clustering *vector data* is traditionally done by using all attributes of the feature space. Density-based techniques [8,11] have shown their strength in contrast to other full-space clustering approaches like k-means. They do not require the number of clusters as an input parameter and are able to find arbitrarily shaped clusters. However, full-space clustering does not scale to high dimensional data since locally irrelevant dimensions obfuscate the clustering structure [4,14]. As a solution, *subspace clustering* methods detect an individual set of relevant dimensions for each cluster [14]. Also for this mining paradigm, density-based clustering approaches [13,3] have resolved the drawbacks of previous subspace cluster definitions. However, none of the introduced subspace clustering techniques considers graph data.

Mining *graph data* can be done in various ways [1]. The task of finding groups of densely connected objects in one large graph, as needed for our method, is often referred to as "graph clustering" or "dense subgraph mining". On overview of the various dense subgraph mining techniques is given in [1]. Furthermore, several different notions of density are used for graph data including cliques,

γ-quasi-cliques [17], and k-cores [5,12]. However, none of the introduced dense subgraph mining techniques considers attribute data annotated to the vertices.

There are some methods considering *graph data and attribute data*. In [6,15] attribute data is only used in a post-processing step. [10] transforms the network into a distance function and afterwards applies traditional clustering. In [18], contrarily, the attribute information is transformed into a graph. [7] extends the k-center problem by requiring that each group has to be a connected subgraph. These approaches [10,18,7] perform full-space clustering on the attributes. [19,20] enriches the graph by further nodes based on the vertices' attribute values and connects them to vertices showing this value. The clustered objects are only pairwise similar and no specific relevant dimensions can be defined.

Recently, two approaches [16,9] were introduced that deal with *subspace clustering and dense subgraph mining*. However, both approaches use too simple cluster definitions. Similar to grid-based subspace clustering [2], a cluster (w.r.t. the attributes) is simply defined by taking all objects located within a given grid cell, i.e. whose attribute values differ by at most a given threshold. The methods are biased towards small clusters with little extend. This drawback is even worsened by considering the used notions of dense subgraphs: e.g. by using quasi-cliques as in [9] the diameter is a priori constrained to a fixed threshold [17]. Very similar objects just slightly located next to a cluster are lost due to such restrictive models. Overall, finding meaningful patterns based on the cluster definitions of [16,9] is questionable. Furthermore, the method in [16] does not eliminate redundancy which usually occurs in the analysis of subspace projections due to the exponential search space containing highly similar clusters.

Our novel model is the first approach using a density-based notion of clusters for the combination of subspace clustering and dense subgraph mining. This allows arbitrary shaped and arbitrary sized clusters hence leading to an unbiased definition of clusters. We remove redundant clusters induced by similar subspace projections resulting in meaningful result sizes.

3 A Density-Based Clustering Model for Combined Data

In this section we introduce our density-based clustering model for the combined clustering of graph data and attribute data. The clusters in our model correspond to dense regions in the graph as well as in the attribute space. For simplicity, we first introduce in Section 3.1 the cluster model for the case that only a single subspace, e.g. the full-space, is considered. The extension to subspace clustering and the definition of a redundancy model to confine the final clustering to the most interesting subspace clusters is introduced in Section 3.2.

Formally, the input for our model is a vertex-labeled graph $G = (V, E, l)$ with vertices V, edges $E \subseteq V \times V$ and a labeling function $l : V \rightarrow \mathbb{R}^D$ where $Dim = \{1, \ldots, D\}$ is the set of dimensions. We assume an undirected graph without self-loops, i.e. $(v, u) \in E \Leftrightarrow (u, v) \in E$ and $(u, u) \notin E$. Furthermore, we use $x[i]$ to refer to the i-th component of a vector $x \in \mathbb{R}^D$.

3.1 Cluster Model for a Single Subspace

In this section we introduce our density-based cluster model for the case of a single subspace. The basic idea of density-based clustering is that clusters correspond to connected dense regions in the dataspace that are separated by sparse regions. Therefore, each clustered object x has to exceed a certain minimal density, i.e. its local neighborhood has to contain a sufficiently high number of objects that are also located in x's cluster. Furthermore, in order to form a cluster, a set of objects has to be *connected* w.r.t. their density, i.e. objects located in the same cluster have to be connected via a chain of objects *from the cluster* such that each object lies within the neighborhood of its predecessor. Consequently, one important aspect of our cluster model is the proper definition of a node's local neighborhood.

If we would simply determine the neighborhood based on the attribute data, as done in [8,11], we could define the attribute neighborhood of a vertex v by its ϵ-*neighborhood*, i.e. the set of all objects which distances to o do not exceed a threshold ϵ. Formally,

$$N_\epsilon^V(v) = \{u \in V \mid dist(l(u), l(v)) \leq \epsilon\}$$

with an appropriate distance function like the maximum norm $dist(x, y) = \max_{i \in \{1,\ldots,D\}} |x[i] - y[i]|$. Just considering attribute data, however, leads to the problem illustrated in Fig. 2. Considering the attribute space, the red triangles form a dense region. (The edges of the graph and the parameter ϵ are depicted in the figure.) Though, this group is not a meaningful cluster since in the graph this vertex set is not densely connected. Accordingly, we have to consider the graph data and attribute data simultaneously to determine the neighborhood.

Intuitively, taking the graph into account can be done by just using adjacent vertices for density computation. The resulting *simple combined neighborhood* of a vertex v would be the intersection

$$N_{\epsilon,adj}^V(v) = N_\epsilon^V(v) \cap \{u \in V \mid (u, v) \in E\}$$

In Fig. 2 the red triangles would not be dense anymore because their simple combined neighborhoods are empty. However, using just the adjacent vertices leads to a too restrictive cluster model, as the next example in Fig. 4 shows.

Assuming that each vertex has to contain 3 objects in its neighborhood (including the object itself) to be dense, we get two densely connected sets, i.e. two clusters, in Fig. 4(a). In Fig. 4(b), we have the same vertex set, the same set of attribute vectors and the same graph density. The example only differs from the first one by the interchange of the attribute values of the vertices v_3 and v_4, which both belong to the same cluster in the first example. Intuitively, this set of vertices should also be a valid cluster in our definition. However, it is not because the neighborhood of v_2 contains just the vertices $\{v_1, v_2\}$. The vertex v_4 is not considered since it is just similar w.r.t. the attributes but not adjacent.

The missing tolerance w.r.t. interchanges of the attribute values is one problem induced by using just adjacent vertices. Furthermore, this approach would not

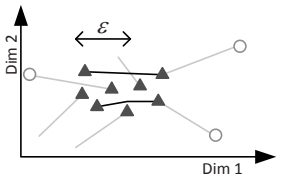


Fig. 2. Dense region in the attribute space but sparse region in the graph

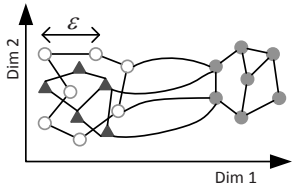
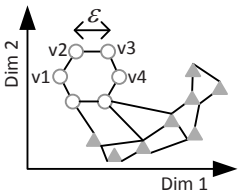
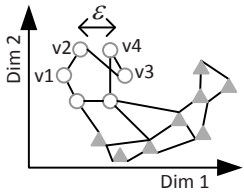


Fig. 3. Novel challenge by using local density computation and k-neighborhoods



(a) Successful with $k \geq \mathbf{1}$, $minPts = 3$



(b) Successful with $k \geq \mathbf{2}$, $minPts = 3$

Fig. 4. Robust cluster detection by using k-neighborhoods. Adjacent vertices ($k = 1$) are not always sufficient for correct detection (left: successful; right: fails)

be tolerant w.r.t. small errors in the edge set. For example in social networks, some friendship links are not present in the current snapshot although the people are aware of each other. Such errors should not prevent a good cluster detection. Thus, in our approach we consider all vertices that are reachable over at most k edges to obtain a more error-tolerant model. Formally, the neighborhood w.r.t. the graph data is given by:

Definition 1 (Graph k-neighborhood). *As vertex u is k-reachable from a vertex v (over a set of vertices V) if*

$$\exists v_1, \ldots, v_k \in V : v_1 = v \wedge v_k = u \wedge \forall i \in \{1, \ldots, k-1\} : (v_i, v_{i+1}) \in E$$

The graph k-neighborhood of a vertex $v \in V$ is given by

$$N_k^V(v) = \{u \in V \mid u \text{ is } x\text{-reachable from } v \text{ (over } V) \wedge x \leq k\} \cup \{v\}$$

Please note that the object v itself is contained in its neighborhood $N_k^V(v)$ as well. Overall, the combined neighborhood of a vertex $v \in V$ considering graph and attribute data can be formalized by intersecting v's graph k-neighborhood with its ϵ-neighborhood.

Definition 2 (Combined local neighborhood). *The combined neighborhood of $v \in V$ is:*

$$N^V(v) = N_k^V(v) \cap N_\epsilon^V(v)$$

Using the combined neighborhood $N^V(v)$ and $k \geq 2$, we get in Fig. 4(a) and Fig. 4(b) the same two clusters. In both examples v_2's neighborhood contains 3 vertices, e.g. $N^V(v_2) = \{v_1, v_2, v_4\}$ in Fig. 4(b). So to speak, we "jump over" the vertex v_3 to find further vertices that are similar to v_2 in the attribute space.

Local density calculation. As mentioned, the "jumping" principle is necessary to get meaningful clusters. However, it leads to a novel challenge not given in previous density-based clustering approaches. Considering Figure 3, the vertices on the right hand side form a combined cluster and are clearly separated from the remaining vertices by their attribute values. However, on the left hand side we have two separate clusters, one consisting of the vertices depicted as dots and one consisting of the vertices depicted as triangles. If we would only consider attribute data for clustering, these two clusters would be merged into one big cluster as the vertices are very similar w.r.t. their attribute values. If we would just use adjacent vertices, this big cluster would not be valid as there are no edges between the dot vertices and the triangle vertices. However, since we allow "jumps" over vertices, the two clusters could be merged, e.g. for $k = 2$.

This problem arises because the dots and triangles are connected via the vertices on the right hand side, i.e. we "jump" over vertices that actually do not belong to the final cluster. Thus, a "jump" has to be restricted to the objects of the same cluster. In Fig. 4(b) e.g., the vertices $\{v_2, v_3, v_4\}$ belong to the same cluster. Thus, reaching v_4 over v_3 to increase the density of v_2 is meaningful. Formally, we have to restrict the k-reachability used in Def. 1 to the vertices contained in the cluster O. In this case, the vertices on the right hand side of Fig. 3 cannot be used for "jumping" and thus the dots are not in the triangles' neighborhoods and vice versa. Thus, we are able to separate the two clusters.

Overall, for computing the neighborhoods and hence the densities, only vertices $v \in O$ of the *same* cluster O can be used. While previous clustering approaches calculate the densities w.r.t. the whole database (all objects in V), our model calculates the densities within the clusters (objects in O). Instead of calculating *global densities*, we determine *local densities* based on the cluster O. While this sounds nice in theory, it is difficult to solve in practice, as obviously the set of clustered objects O is not known a priori but has to be determined. The cluster O depends on the density values of the objects, while the density values depend on O. *So we get a cyclic dependency of both properties.*

In our theoretical clustering model, we can solve this cyclic dependency by assuming a set of clustered objects O as given. The algorithmic solution is presented in Sec. 4. Formally, given the set of clustered objects O three properties have to be fulfilled: First, each vertex $v \in O$ has to be dense w.r.t. it local neighborhood. Second, the spanned region of these objects has to be (locally) connected since otherwise we would have more than two clusters (cf. Fig 3). Last, the set O has to be maximal w.r.t. the previous properties since otherwise some vertices of the cluster are lost. Overall,

Definition 3 (Density-based combined cluster). *A combined cluster in a graph $G = (V, E, l)$ w.r.t. the parameters k, ϵ and $minPts$ is a set of vertices $O \subseteq V$ that fulfills the following properties:*

(1) high local density: $\forall v \in O : |N^O(v)| \geq minPts$
(2) locally connected: $\forall u, v \in O : \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : w_i \in N^O(w_{i+1})$
(3) maximality: $\neg \exists O' \supset O : O'$ *fulfills (1) and (2)*

Please note that the neighborhood calculation $N^O(v)$ is always done w.r.t. to the set O and not w.r.t. the whole graph V. Based on this definition and by using $minPts = 3$, $k \geq 2$, we can e.g. detect the three clusters from Fig. 3. By using a too small value for k (as for example $k = 1$ in Fig. 4(b)), clusters are often split or even not detected at all. On the other hand, if we choose k too high, we run the risk that the graph structure is not adequately considered any more. However, for the case depicted in Fig. 3 our model detects the correct clusters even for arbitrary large k values as the cluster on the right hand side is clearly separated from the other clusters by its attribute values. The two clusters on the left hand side are never merged by our model as we do not allow jumps over vertices outside the cluster.

In this section we introduced our combined cluster model for the case of a single subspace. As shown in the examples, the model can detect clusters that are dense in the graph as well as in the attribute space and that can often not be detected by previous approaches.

3.2 Overall Subspace Clustering Model

In this section we extend our cluster model to a subspace clustering model. Besides the adapted cluster definition we have to take care of redundancy problems due to the exponential many subspace projections. As mentioned in the last section, we are using the maximum norm in the attribute space. If we just want to analyze subspace projections, we can simply define the maximum norm restricted to a subspace $S \subseteq Dim$ as

$$dist^S(x, y) = \max_{i \in S} |x[i] - y[i]|$$

In principle any L_p norm can be restricted in this way and can be used within our model. Based on this distance function, we can define a subspace cluster which fulfills the cluster properties just in a subset of the dimensions:

Definition 4 (Density-based combined subspace cluster). *A combined subspace cluster $C = (O, S)$ in a graph $G = (V, E, l)$ consists of a set of vertices $O \subseteq V$ and a set of relevant dimensions $S \subseteq Dim$ such that O forms a combined cluster (cf. Def 3) w.r.t. the local subspace neighborhood $N_S^O(v) = N_k^O(v) \cap N_{\epsilon,S}^O(v)$ with $N_{\epsilon,S}^O(v) = \{u \in O \mid dist^S(l(u), l(v)) \leq \epsilon\}$.*

As we can show, our subspace clusters have the anti-monotonicity property: For a subspace cluster $C = (O, S)$, for every $S' \subseteq S$ there exists a vertex set $O' \supseteq O$ such that (O', S') is a valid cluster. This property is used in our algorithm to find the valid clusters more efficiently.

Proof. For every two subspaces S, S' with $S' \subseteq S$ and every pair of vertices u, v it holds that $dist^{S'}(l(u), l(v)) \leq dist^S(l(u), l(v))$. Thus for every vertex $v \in O$ we get $N_{S'}^O(v) \supseteq N_S^O(v)$. Accordingly, the properties (1) and (2) from Def. 3 are fulfilled by (O, S'). If (O, S') is maximal w.r.t. these properties, then (O, S') is a valid combined subspace cluster. Else, by definition there exists a vertex set $O' \supset O$ such that (O', S') is a valid subspace cluster.

Redundancy removal. Because of the density-based subspace cluster model, there can not be an overlap between clusters in the same subspace. However, a vertex can belong to several subspace clusters in different subspaces, thus clusters from different subspaces can overlap. Due to the exponential number of possible subspace projections, an overwhelming number of (very similar) subspace clusters may exist. Whereas allowing overlapping clusters in general makes sense in many applications, allowing too much overlap can lead to highly redundant information. Thus, for meaningful interpretation of the result, removing redundant clusters is crucial. Instead of simply reporting the highest dimensional subspace clusters or the cluster with maximal size, we use a more sophisticated redundancy model to confine the final clustering to the most interesting clusters.

However, defining the interestingness of a subspace cluster is a non-trivial task since we have two important properties "number of vertices" and "dimensionality". Obviously, optimizing both measures simultaneously is not possible as clusters with higher dimensionality usually consist of fewer vertices than their low-dimensional counterparts (cf. anti-monotonicity). Thus we have to realize a trade-off between the size and the dimensionality of a cluster. We define the interestingness of a single cluster as follows:

Definition 5 (Interestingness of a cluster). *The interestingness of a combined subspace cluster $C = (O, S)$ is computed as*

$$Q(C) = |O| \cdot |S|$$

For the selection of the final clustering based on this interestingness definition we use the redundancy model introduced by [9]. This model defines a binary redundancy relation between two clusters as follows:

Definition 6 (Redundancy between clusters). *Given the redundancy parameters $r_{obj}, r_{dim} \in [0, 1]$, the binary redundancy relation $\prec_{red}$ is defined by:*

$$\text{For all combined clusters } C = (O, S), \overline{C} = (\overline{O}, \overline{S}):$$
$$C \prec_{red} \overline{C} \Leftrightarrow Q(C) < Q(\overline{C}) \wedge \frac{|O \cap \overline{O}|}{|O|} \geq r_{obj} \wedge \frac{|S \cap \overline{S}|}{|S|} \geq r_{dim}$$

Using this relation, we consider a cluster C as redundant w.r.t. another cluster $\overline{C}$ if C's quality is lower and the overlap of the clusters' vertices and dimensions exceeds a certain threshold. It is crucial to require both overlaps for the redundancy. E.g. two clusters that contain similar vertices, but lie in totally different dimensions represent different information and thus should both be considered in the output.

Please note that the redundancy relation $\prec_{red}$ is non-transitive, i.e. we cannot just discard every cluster that is redundant w.r.t. any other cluster. Therefore we have to select the final clustering such that it does not contain two clusters that are redundant w.r.t. each other. At the same time, the result set has to be maximal with this property, i.e. we can not just leave out a cluster that is non-redundant. Overall, the final clustering has to fulfill the properties:

Definition 7 (Optimal density-based combined subspace clustering). *Given the set of all combined clusters $Clusters$, the optimal combined clustering $Result \subseteq Clusters$ fulfills*

- *redundancy-free property:* $\neg\exists C_i, C_j \in Result : C_i \prec_{red} C_j$
- *maximality property:* $\forall C_i \in Clusters \backslash Result : \exists C_j \in Result : C_i \prec_{red} C_j$

Our clustering model enables us to detect arbitrarily shaped subspace clusters based on attribute and graph densities without generating redundant results.

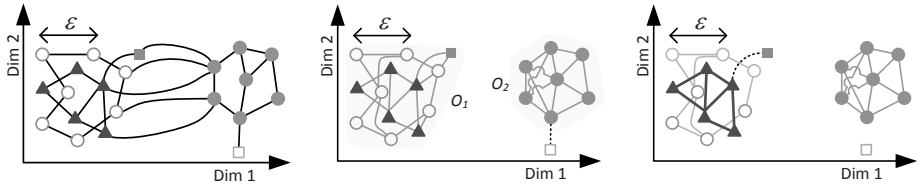
4 The DB-CSC Algorithm

In the following section we describe the DB-CSC (**D**ensity-**B**ased **C**ombined **S**ubspace **C**lustering) algorithm for detecting the optimal clustering result. In Section 4.1 we present the detection of our density-based clusters in a single subspace S and in Section 4.2 we introduce the overall processing scheme using different pruning techniques to enhance the efficiency.

4.1 Finding Clusters in a Single Subspace

To detect the clusters in a single subspace S, we first introduce a graph transformation that represents attribute and graph information simultaneously.

Definition 8 (Enriched subgraph). *Given a set of vertices $O \subseteq V$, a subspace S, and the original graph $G = (V, E, l)$, the enriched subgraph $G_S^O = (V', E')$ is defined by $V' = O$ and $E' = \{(u, v) \mid v \in N_S^O(u) \wedge v \neq u\}$ using the distance function $dist^S$.*



(a) Original graph (b) Enriched graph G_S^V (c) $G_S^{O_i}$ for subsets O_i

Fig. 5. Finding clusters by ($minPts-1$)-cores in the enriched graphs (k=2, $minPts$=3)

Two vertices are adjacent in the enriched subgraph iff their attribute values are similar in S and the vertices are connected by at most k edges in the original graph (using just vertices from O). In Fig. 5(b) the enriched subgraph for the whole set of vertices V is computed while Fig. 5(c) just considers the subset O_1 and O_2 respectively. In this graph we concentrate on the detection of ($minPts-1$)-cores, which are defined as maximal connected subgraphs $O_i \subseteq V$ in which all vertices have at least a degree of ($minPts-1$). We show:

Theorem 1 (Equivalence of representations). *Let $O \subseteq V$ a set of vertices and $S \subseteq Dim$ a subspace. $C = (O, S)$ fulfills property (1) and (2) of Definition 3 if and only if the enriched subgraph $G_S^O = (O, E')$ contains* ***a single*** *($minPts-1$)-core that covers all vertices O.*

Proof. $*$ For property (1) of Def. 3 we get: high local density $\Leftrightarrow \forall v \in O : |N^O(v)| \geq minPts \Leftrightarrow \forall v \in O : |N^O(v) \backslash \{v\}| \geq minPts - 1 \Leftrightarrow \forall v \in O : deg_{E'}(v) \geq minPts - 1 \Leftrightarrow$ minimal vertex degree of $minPts - 1$ in G_S^O
$*$ For property (2) of Def. 3 we get: locally connected $\Leftrightarrow \forall u, v \in O$: $\exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : w_i \in N_S^O(w_{i+1}) \Leftrightarrow \forall u, v \in O : \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : (w_i, w_{i+1}) \in E' \Leftrightarrow G_S^O$ is connected

The theorem implies that our algorithm only has to analyze vertices that potentially lead to ($minPts-1$)-cores. The important observation is: If G_S^O is a ($minPts-1$)-core, then for each graph $G_S^{O'}$ with $O' \supseteq O$ the set O will also be contained in a ($minPts-1$)-core. (This holds since $N_S^{O'}(u) \supseteq N_S^O(u)$ and hence $G_S^{O'}$ contains all edges of G_S^O.) Thus, each potential cluster (O, S) especially has to be contained within a ($minPts-1$)-core of the graph G_S^V. Overall, we first extract from G_S^V all ($minPts-1$)-cores, since only these sets could lead to valid clusters. In Fig. 5(b) these sets are highlighted.

However, keep in mind that not all ($minPts-1$)-cores correspond to valid clusters. Theorem 1 requires that *a single* ($minPts-1$)-core covering all vertices is induced by the enriched subgraph. Figure 5(b) for example contains two cores. As already discussed, the left set O_1 is not a valid cluster but has to be split up.

Thus, if the graph G_S^V contains *a single* ($minPts-1$)-core O_1 with $O_1 = V$ we get a valid cluster and the cluster detection is finished in this subspace. In the other cases, however, we recursively have to repeat the procedure for each ($minPts-1$)-core $\{O_1, \ldots, O_m\}$ detected in G_S^V , i.e. we determine the smaller graphs $G_S^{O_i}$ and their contained ($minPts-1$)-cores. Since in each step the (maximal) ($minPts-1$)-cores are analyzed and refined, we ensure besides property (1) and (2) – due to Theorem 1 – also property (3) of Definition 3.

Formally, the set of resulting clusters $Clus = \{C_1, \ldots, C_m\}$ corresponds to a fixpoint of the function

$$f(Clus) = \{(O', S) \mid O' \text{ is a } (minPts-1)\text{-core in } G_S^{O_i} \text{ with } C_i = (O_i, S) \in Clus\}$$

This fixpoint can be reached by $f(f(\ldots f(\{(V, S)\}))) = Clus$. Overall, this procedures enables us to detect all combined clusters in the subspace S.

4.2 Finding Clusters in Different Subspaces

This chapter describes how we efficiently determine the clusters located in different subspaces. In principle, our algorithm has to analyze each subspace. We enumerate these subspaces by a depth first traversal through the subspace lattice. To avoid enumerating the same subspace several times we assume an order $d_1, d_2, \ldots, d_D$ on the dimensions. We denote the dimension with the highest index in subspace S by $\max\{S\}$ and extend the subspace only by dimensions that are ordered behind $\max\{S\}$. This principle has several advantages:

By using a depth first search, the subspace S is analyzed before the subspace $S' = S \cup \{d\}$ for $d > \max\{S\}$. Based on the anti-monotonicity we know that

each cluster in S' has to be a subset of a cluster in S. Thus, in subspace S' we do not have to start with the enriched subgraph G_S^V but it is sufficient to start with the vertex sets of the known clusters, i.e. if the clusters of subspace S are given by $Clus = \{C_1, \ldots, C_m\}$, we will determine the fixpoint $f(f(\ldots f(\{(O_i, S')\})))$ for each cluster $C_i = (O_i, S) \in Clus$. This is far more efficient since the vertex sets are smaller. In Fig. 6 the depth first traversal based on the clusters of the previous subspace is shown in line 21, 23 and 29. The actual detection of clusters based on the vertices O is realized in line 13-19, which corresponds to the fixpoint iteration described in the previous section.

Using a depth first search enables us to store a set of parent clusters (beforehand detected in lower dimensional subspaces) that a new cluster is based on (cf. line 22). Furthermore, given a set of vertices O in the subspace S we know that by traversing the current subtree only clusters of the kind $C_{reach} = (O_{reach}, S_{reach})$ with $O_{reach} \subseteq O$ and $S \subseteq S_{reach} \subseteq S \cup \{\max\{S\}+1, \ldots, D\}$ can be detected. This information together with the redundancy model allows a further speed-up of the algorithm. The overall aim is to stop the traversal of a subtree if each of the reachable (potential) clusters is redundant to one parent cluster, i.e. if there exists $C \in Parents$ such that $C_{reach} \prec_{red} C$ for each C_{reach}. Traversing such a subtree is not worthwhile since the contained clusters are probably excluded from the result later on due to their redundancy.

Redundancy of a subtree occurs if the three properties introduced in Def. 6 hold. The second property (object overlap) is always fulfilled since each O_{reach} is a subset of any cluster from $Parents$ (cf. anti-monotonicity). The maximal possible quality of the clusters C_{reach} can be estimated by $Q_{max} = |O| \cdot |S \cup \{\max\{S\}+1, \ldots, D\}|$. By focusing on the clusters $C_p = (O_p, S_p) \in Parents$ with $Q_{max} < Q(C_p)$ we ensure the first redundancy property. The third property (dimension overlap) is ensured if $|S_p| \geq |S \cup \{\max\{S\}+1, \ldots, D\}| \cdot r_{dim}$ holds. In this case we get for each C_{reach}: $|S_p| \geq |S_{reach}| \cdot r_{dim} \Leftrightarrow |S_p \cap S_{reach}| \geq |S_{reach}| \cdot r_{dim} \Leftrightarrow \frac{|S_p \cap S_{reach}|}{|S_{reach}|} \geq r_{dim}$. Those parent clusters fulfilling all three properties are stored within $Parents_{red}$ (line 24).

If $Parents_{red}$ is empty, we have to traverse the subtree (else case, line 28). If it is not empty (line 25), the current subtree is redundant to at least one parent cluster. We currently stop traversing this subtree. However, we must not directly prune the subtree ST: if the clusters from $Parents_{red}$ themselves are *not* included in the result, clusters from the subtree would become interesting again. Thus, we do not finally reject the subtree ST but we store the required information and add the subtree to a priority queue.

Processing this queue is the core of the DB-CSC algorithm. The priority queue contains clusters (line 6, 20) and non-traversed subtrees (line 9, 27). We successively take the object with the highest (estimated) quality from the queue. If it is a cluster that is non-redundant to the current result, we add it to the result (line 7-8). If it is a subtree, we check if some cluster from $Parents_{red}$ is already included in the result: if so, we finally reject this subtree (line 10); otherwise, we have to restart traversing this subtree (line 11).

```
method: main()
1  Result = ∅ // current result set
2  queue = ∅ // priority queue with clusters and subtrees, descendingly sorted by quality
3  for d ∈ Dim do DFS_traversal({d}, V, ∅)
4  while queue ≠ ∅ do
5      remove first (highest-quality) object Obj from queue
6      if Obj is cluster then // check redundancy
7          for C ∈ Result do if( Obj ≺_red C) goto line 4 // discard redundant cluster
8          Result = Result ∪ {Obj} // cluster is non-redundant
9      else // Obj is subtree ST = (S, O, Q_max, Parents, Parents_red)
10         if Parents_red ∩ Result ≠ ∅ then goto line 4 // discard whole subtree
11         else DFS_traversal(S, O, Parents) // subtree is non-redundant, restart traversal
12 return Result

method: DFS_traversal(subspace S, candidate vertices O, parent clusters Parents)
13 foundClusters = ∅, prelimClusters = {O}
14 while prelimClusters ≠ ∅ do
15     remove first candidate O_x from prelimClusters
16     generate enriched subgraph G_S^{O_x}
17     determine (minPts − 1)-cores → Cores = {O'_1, ..., O'_m}
18     if |Cores| = 1 ∧ O'_1 = O_x then foundClusters = foundClusters ∪ {(O'_1, S)}
19     else prelimClusters = prelimClusters ∪ Cores
20 add foundClusters to queue
21 for C_i = (O_i, S) ∈ foundClusters do
22     Parents_i = Parents ∪ {C_i}
23     for d ∈ {max{S} + 1, ..., D} do
24         determine Parents_red ⊆ Parents_i to which whole subtree is redundant
25         if Parents_red ≠ ∅ then
26             calc. subtree information ST = (S ∪ {d}, O_i, Q_max, Parents_i, Parents_red)
27             add ST to queue // (currently) do not traverse subtree!
28         else
29             DFS_traversal(S ∪ {d}, O_i, Parents_i) // check only subsets of O_i
```

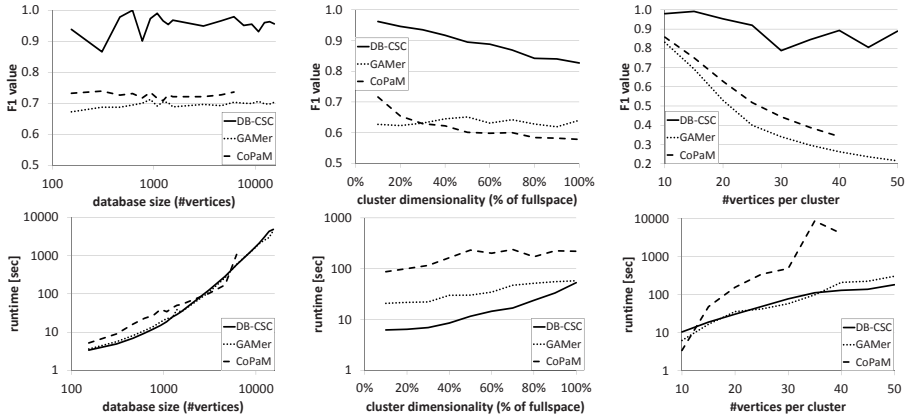
Fig. 6. DB-CSC algorithm

Overall, our algorithm efficiently determines the optimal clustering solution because only small vertex sets are analyzed for clusters and whole subtrees (i.e. sets of clusters) are pruned using the redundancy model.

5 Experimental Evaluation

Setup. We compare DB-CSC to GAMer [9] and CoPaM [16], two approaches that combine subspace clustering and dense subgraph mining. In our experiments we use real world data sets as well as synthetic data. By default the synthetic datasets have 20 attribute dimensions and contain 80 combined clusters each with 15 nodes and 5 relevant dimensions. Additionally we add random nodes and edges to represent noise in the data. The clustering quality is measured by the F1 measure [9], which compares the detected clusters to the "hidden" clusters. The efficiency is measured by the algorithms' runtime.

Varying characteristics of the data. In the first experiment (Fig. 7(a)) we vary the database size of our synthetic datasets by varying the number of generated combined clusters. The runtime of all algorithms increases with increasing database size (please note the logarithmic scale on both axes). For the datasets

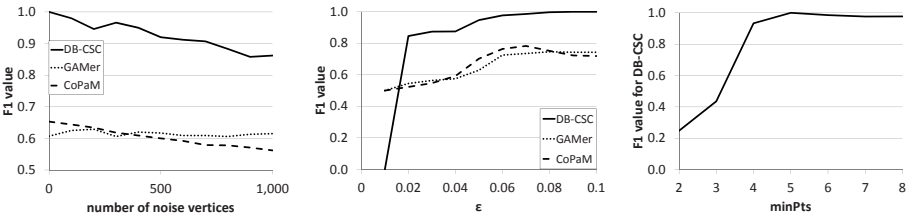


(a) Varying database size (b) Varying dimensionality (c) Varying cluster size

Fig. 7. Quality (top row) and Runtime (bottom row) w.r.t. varying data characteristics

with more than 7000 vertices, CoPaM is not applicable any more due to heap overflows (4GB). While the runtimes of the different algorithms are very similar, in terms of clustering quality DB-CSC obtains significantly better results than the other approaches. The competing approaches tend to output only subsets of the hidden clusters due to their restrictive cluster models. In the next experiment (Fig. 7(b)) we vary the dimensionality of the hidden clusters. The runtime of all algorithms increases for higher dimensional clusters. The clustering qualities of DB-CSC and CoPaM slightly decrease. This can be explained by the fact that for high dimensional clusters it is likely that additional clusters occur in subsets of the dimensions. However, DB-CSC still has the best clustering quality and runtime in this experiment. In Fig. 7(c) the cluster size (i.e. the number of vertices per cluster) is varied. The runtimes of DB-CSC and GAMer are very similar to each other, whereas the runtime of CoPaM increases dramatically until it is not applicable any more. The clustering quality of DB-CSC remains relatively stable while the qualites of the other approaches decrease constantly. For increasing cluster sizes the expansion of the clusters in the graph as well as in the attribute space increases, thus the restrictive cluster models of GAMer and CoPaM can only detect subsets of them.

Robustness. In Fig. 8(a) we analyze the robustness of the methods w.r.t. the number of "noise" vertices in the datasets. The clustering quality of all approaches decreases for noisy data, however the quality of DB-CSC is still reasonably high even for 1000 noise vertices (which is nearly 50% of the overall dataset). In the next experiment (Fig. 8(b)) we vary the clustering parameter ϵ. For GAMer and CoPaM we vary the allowed width of a cluster in the attribute space instead of ϵ. As shown in the figure, by choosing ϵ too small we cannot find all clusters and thus get smaller clustering qualities. However, for $\epsilon > 0.05$ the clustering quality of DB-CSC remains stable. The competing methods have lower quality. In the last experiment (Fig. 8(c)) we evaluate the robustness of



(a) Quality vs. noise (b) Quality vs. ϵ (c) Quality vs. $minPts$

Fig. 8. Robustness of the methods w.r.t. noise and parameter values

DB-CSC w.r.t. the parameter $minPts$. For too small values for $minPts$, many vertex sets are falsely detected as clusters, thus we obtain small clustering qualities. However, for sufficiently high $minPts$ values the quality remains relatively stable, similar to the previous experiment.

Overall, the experiments show that DB-CSC obtains significantly higher clustering qualities. Even though it uses a more sophisticated cluster model than GAMer and CoPaM, the runtimes of DB-CSC are comparable to (and in some cases even better than) those of the other approaches.

Real world data. As real world data sets we use gene data[1] and patent data[2] as also used in [9]. Since for real world data there are no "hidden" clusters given that we could compare our clustering results with, we compare the properties of the clusters found by the different methods. For the gene data DB-CSC detects 9 clusters with a mean size of 6.3 and a mean dimensionality of 13.2. In contrast, GAMer detects 30 clusters (mean size: 8.8 vertices, mean dim.: 15.5) and CoPaM 115581 clusters (mean size: 9.7 vertices, mean dim.: 12.2), which are far too many to be interpretable. In the patent data, DB-CSC detects 17 clusters with a mean size of 19.2 vertices and a mean dimensionality of 3. In contrast, GAMer detects 574 clusters with a mean size of 11.7 vertices and a mean dimensionality of 3. CoPaM did not finish on this dataset within two days. The clusters detected by DB-CSC are more expanded than the clusters of GAMer, which often simply are subsets of the clusters detected by DB-CSC.

6 Conclusion

We introduce the combined clustering model which simultaneously considers graph data and attribute data in subspaces. Our novel model is the first approach that exploits the advantages of density-based clustering in both domains. Based on the novel notion of local densities, our clusters correspond to dense regions in the graph as well as in the attribute space. To avoid redundancy in the result, our model selects only the most interesting clusters for the final clustering. We develop the algorithm DB-CSC to efficiently determine the combined clustering

[1] `http://thebiogrid.org/` and `http://genomebiology.com/2005/6/3/R22`
[2] `http://www.nber.org/patents/`

solution. The clustering quality and the efficiency of DB-CSC are demonstrated in the experimental section.

Acknowledgment. This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany, and the B-IT Research School.

References

1. Aggarwal, C., Wang, H.: Managing and Mining Graph Data. Springer, New York (2010)
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD, pp. 94–105 (1998)
3. Assent, I., Krieger, R., Müller, E., Seidl, T.: EDSC: Efficient density-based subspace clustering. In: CIKM, pp. 1093–1102 (2008)
4. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: ICDT, pp. 217–235 (1999)
5. Dorogovtsev, S., Goltsev, A., Mendes, J.: K-core organization of complex networks. Physical Review Letters 96(4), 40601 (2006)
6. Du, N., Wu, B., Pei, X., Wang, B., Xu, L.: Community detection in large-scale social networks. In: WebKDD/SNA-KDD, pp. 16–25 (2007)
7. Ester, M., Ge, R., Gao, B.J., Hu, Z., Ben-Moshe, B.: Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In: SDM (2006)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, pp. 226–231 (1996)
9. Günnemann, S., Färber, I., Boden, B., Seidl, T.: Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In: ICDM, pp. 845–850 (2010)
10. Hanisch, D., Zien, A., Zimmer, R., Lengauer, T.: Co-clustering of biological networks and gene expression data. Bioinformatics 18, 145–154 (2002)
11. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: KDD, pp. 58–65 (1998)
12. Janson, S., Luczak, M.: A simple solution to the k-core problem. Random Structures & Algorithms 30(1-2), 50–62 (2007)
13. Kailing, K., Kriegel, H.P., Kroeger, P.: Density-connected subspace clustering for high-dimensional data. In: SDM, pp. 246–257 (2004)
14. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. TKDD 3(1), 1–58 (2009)
15. Kubica, J., Moore, A.W., Schneider, J.G.: Tractable group detection on large link data sets. In: ICDM, pp. 573–576 (2003)
16. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM, pp. 593–604 (2009)
17. Pei, J., Jiang, D., Zhang, A.: On mining cross-graph quasi-cliques. In: KDD, pp. 228–238 (2005)
18. Ulitsky, I., Shamir, R.: Identification of functional modules using network topology and high-throughput data. BMC Systems Biology 1(1) (2007)
19. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB 2(1), 718–729 (2009)
20. Zhou, Y., Cheng, H., Yu, J.X.: Clustering large attributed graphs: An efficient incremental approach. In: ICDM, pp. 689–698 (2010)