Density-Based Subspace Clustering in Heterogeneous Networks

Brigitte Boden[1], Martin Ester[2], and Thomas Seidl[1]

[1] RWTH Aachen University, Aachen, Germany
{boden,seidl}@cs.rwth-aachen.de
[2] Simon Fraser University, Burnaby, BC, Canada
ester@cs.sfu.ca

Abstract. Many real-world data sets, like data from social media or bibliographic data, can be represented as heterogeneous networks with several vertex types. Often additional attributes are available for the vertices, such as keywords for a paper. Clustering vertices in such networks, and analyzing the complex interactions between clusters of different types, can provide useful insights into the structure of the data. To exploit the full information content of the data, clustering approaches should consider the connections in the network as well as the vertex attributes. We propose the density-based clustering model TCSC for the detection of clusters in heterogeneous networks that are densely connected in the network as well as in the attribute space. Unlike previous approaches for clustering heterogeneous networks, TCSC enables the detection of clusters that show similarity only in a subset of the attributes, which is more effective in the presence of a large number of attributes.

1 Introduction

In many applications, data of various kinds are available, and there is a need for analyzing such data. Clustering, the task of grouping objects based on their similarity, is one of the most important data mining tasks, and clustering algorithms for different kinds of data exist. Graph clustering aims at grouping the vertices of a network into clusters such that many edges between vertices of the same cluster exist, i.e. the vertices are densely connected. While most graph clustering methods are constrained to homogeneous networks (networks with a single vertex type), real-world data can often better be represented by *heterogeneous* networks with several vertex types. For example, bibliographic data can be represented as a network with the vertex types "paper" and "author". When we consider heterogeneous networks, a novel challenge for clustering arises: Besides detecting clusters, clustering approaches should also analyze the interactions between clusters of different types, e.g. "which groups of authors are interested in which groups of papers?" Furthermore, real-world data often contains additional information ("attributes") about the vertices of a graph. E.g., a "paper" vertex can be further described by a vector of keywords. To exploit the full information content of the data, the similarity of the vertex attributes should be considered

T. Calders et al. (Eds.): ECML PKDD 2014, Part I, LNCS 8724, pp. 149–164, 2014.

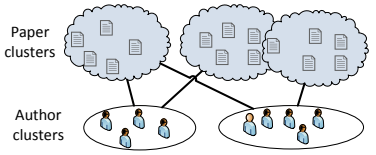


Fig. 1. Example author-paper clustering

for the clustering, as well as the connections in the network. An important aspect is that not all of the attributes may be relevant for a cluster. E.g., for a cluster of papers on similar topics, we would expect the papers to have *some*, but not *all* keywords in common. Thus, we aim at detecting clusters of vertices in heterogeneous networks that are densely connected and also show similarity in a subset of the attributes (called *subspace*), similar to the principle of *subspace clustering* for vector data [8]. To the best of our knowledge, there exists no previous approach for subspace clustering in heterogeneous networks.

In principle, it would be possible to project the network to a homogeneous network just containing vertices of one of the types (e.g. build a co-author network by connecting authors with common papers). However, by doing this information about the other types (e.g. the topics of the papers) is lost. Furthermore, an important aspect in our work is analyzing the connections between clusters of different types, which would not be possible at all in such a setting. In our experimental section, we show the superiority of our approach over a baseline using such a projection.

In our work, we consider heterogeneous networks that contain edges between vertices of different types (e.g. a paper is connected to its authors), but can also contain edges between vertices of the same type (e.g. citations between papers). Furthermore, for each of the vertex types there can be additional attributes. In such networks, we want to cluster the vertices of each type such that the clusters of different types interact with each other. An important challenge is how to model the interactions between the clusters. Intuitively, two clusters (of different types) are connected if the vertices of each cluster are densely connected *via the vertices of the other cluster*. Consider the example in Fig. 1. Here we observe two author clusters and three paper clusters (two of which are overlapping). The connections between the clusters indicate that the vertices of those clusters are connected by many edges. Naturally, a group of authors can publish papers about different topics, and also different groups of authors publish papers on the same topic. Thus it makes sense that an author cluster can be connected to several paper clusters and vice versa. Each cluster can interact with a different number of clusters. Therefore, just connecting each cluster to a specified number of other clusters would be problematic. Thus, in our approach the number of connections of a cluster is not restricted.

We observe that there are different ways to represent a data set as a heterogeneous network. Information about entities (e.g. words contained in papers) can be modeled in different ways: Either as an additional vertex type or as an attribute of another vertex type. In our work, we model only those informa-

tion types as vertex types that we want to cluster. Other types of information are modeled as attributes. Furthermore, we want to highlight that there is no unique definition of the clustering problem in heterogeneous networks. The existing approaches vary greatly in their clustering objectives. Some approaches (e.g. [15]) cluster only the vertices of one type, while the vertices of other types are considered as "attribute types" of the clustered type. Other approaches (e.g. [13]) aim at clustering the vertices of all types such that each cluster contains vertices of different types. In Fig. 1, those approaches would either partition the authors into three clusters or merge two of the paper clusters in order to find a clustering of both types. Other approaches ([16], [17]) partition the vertices of each type separately, with the aim that the group membership of two vertices determines the probability of an edge between them. In this paper[1], we present the cluster model TCSC (Typed Combined Subspace Cluster), which belongs to the last, most general, category. In contrast to the previous approaches, TCSC additionally considers the similarity of vertices in subspaces of their attributes and allows the clusters to overlap, which makes sense in many applications. However, redundancy in the clustering result due to too much overlap is avoided by using a redundancy model. We introduce the algorithm HSC (Heterogeneous Subspace Clustering) for detecting TCSC clusters and evaluate it in experiments on real-world data sets.

2 Related Work

Combined Clustering of Graph and Attribute Data. Recently, several clustering approaches have been proposed that consider (homogeneous) graphs with vertex attributes. These approaches can be seen as a combination of graph clustering and vector clustering approaches. However, they mostly rely on fullspace-clustering on the vertex attributes (e.g. [12,20,19]) or only consider binary attributes [1]. The approaches [10,6,7] propose the combination of subspace clustering and graph clustering, aiming at finding clusters of vertices that are densely connected and as well show similar attribute values in a subset of their attributes. However, none of them considers heterogeneous networks.

Clustering in Heterogeneous Networks. The existing approaches for clustering in heterogeneous networks vary greatly in the types of networks that they consider, as well as in their clustering objectives. Several approaches [4,18] consider graphs with a single vertex type and multiple edge types. In some cases, such networks are called multi-dimensional [18] or multislice [11] networks. [4] considers graphs with multiple edge types and edge attributes. In such graphs, densely connected clusters are detected that also have similar attribute values. Other approaches [2,16] consider bipartite graphs: [2] defines a null model for modularity which considers bipartite networks and detects communities based on this measure. [16] proposes a new modularity measure for bipartite graphs, resulting in one partition of the vertices for each vertex type.

[1] The contents of this paper are also included in the first author's PhD thesis [3].

There also exist approaches that can handle graphs with an arbitrary number of vertex types: [5,15] cluster star-structured heterogeneous networks with one central type, where only the clustering of the vertices of the central type is optimized. In [14], networks with several vertex types are considered, which are not restricted to star-structured networks. The user has to specify a target type, i.e. a vertex type that should be clustered. The other types are called "feature types" and are used like attributes of the target type vertices. [13] considers a heterogeneous network with incomplete vertex attributes. The authors mention that only a subset of the attributes may be relevant for the clustering (similar to the idea of subspace clustering). However, in this approach the user has to specify the relevant set of attributes. In the resulting clustering, each cluster can contain vertices of every type. The clustering is mostly based on the attributes, while the links are only used to ensure a "structural consistency" (i.e. connected vertices are clustered together with higher probability). However, the resulting clusters do not have to be dense or even connected in the network. [9] propose a random-walk based approach for community detection in heterogeneous networks, which aims at finding a single community based on a set of seed items.

The most similar approach to our work is [17]. This approach considers evolving multi-mode networks, i.e. networks with different types of vertices that evolve over time. The vertices of each type are clustered simultaneously, with the aim that the group membership of two vertices determines their interaction. However, the approach does not provide information about connections between groups. Furthermore, clusters should evolve smoothly over the time steps. The proposed method only considers multi-partite networks, but extensions for considering edges between vertices of the same type and vertex attributes are mentioned. In our experimental section, we compare this approach (with the extension for using attributes) with our approach.

3 The TCSC Clustering Model

In this section, we introduce our TCSC model for clustering in heterogeneous networks. Basically, a cluster consists of a set of vertices of the same type that are densely connected via the vertices of the other types and show similar attribute values in a subset of their dimensions (this subset is called the *subspace* of the cluster). The model is partly based on the DB-CSC model [7] for homogeneous networks with vertex attributes. A new important challenge for heterogeneous networks is the detection of interactions between the clusters of different types.

For defining the clusters, we adopt the principle of density-based clustering, which allows the detection of dense clusters without restricting them to certain shapes or sizes. Basically, in density-based clustering clusters are defined as dense regions in the data space that are separated by sparse regions. In our work, we aim at detecting clusters that are not only dense considering their attribute values, but also are densely connected in the network via the vertices of their interacting clusters of other types. Thus, the clusters in our model correspond

to dense regions in the graph as well as in a subspace of the attribute space. Therefore, we define the local neighborhood of a vertex such that it represents the graph neighborhood as well as the neighborhood in the attribute space.

Formally, the input for our model is a vertex-labeled graph with T different vertex types. Formally, $G = (V, E, t, l)$ with vertices V, edges E, a type indicator function $t : V \to \{1, \ldots, T\}$ and a vertex labeling function l. Let V_i denote the set of vertices of type i: $V_i = \{v \in V : t(v) = i\}$ and Dim_i the set of dimensions for type i, then $l : V_i \to \mathbb{R}^{|Dim_i|}$.

Neighborhood Definitions. For the clustering, we do not only consider vertices as neighbors that are directly connected by an edge, but also vertices that are connected via other vertices. For example, in a paper-author network we would consider two authors as neighbors if they are co-authors of a common paper, i.e. they are connected via two hops in the network. Therefore, we use the k-neighborhood of a vertex to define its local neighborhood in the graph. Formally, the graph k-neighborhood is defined as follows:

Definition 1 (Graph k-neighborhood). *A vertex u is k-reachable from a vertex v (over a set of vertices V) if $\exists v_1, \ldots, v_k \in V : v_1 = v \wedge v_k = u \wedge \forall j \in \{1, \ldots, k-1\} : (v_j, v_{j+1}) \in E$. The graph k-neighborhood of vertex $v \in V$ is given by: $N_k^V(v) = \{u \in V \mid u$ is j-reachable from v (over $V) \wedge j \leq k\} \cup \{v\}$.*

Furthermore, we define the ϵ-neighborhood of a vertex in the attribute space. Naturally, this neighborhood can only contain vertices of the same type:

Definition 2 (ϵ-neighborhood). *The distance between two vertices x and y of type i w.r.t. a subspace $S \subseteq Dim_i$ is defined as the maximum norm[2] $dist^S(x, y) = \max_{d \in S} |x[d] - y[d]|$ with the special case $dist^{\emptyset}(x, y) = 0$. The ϵ-neighborhood of $v \in V$ for a subspace $S \subseteq Dim_{t(v)}$ is defined as: $N_{\epsilon,S}^V(v) = \{u \in V_{t(v)} \mid dist^S(l(u), l(v)) \leq \epsilon\}$*

As we want to consider the connections in the graph and the similarity in the attribute space simultaneously, we define a combined local neighborhood:[3]

Definition 3 (Combined local neighborhood). *The combined neighborhood of $v \in V$ w.r.t. a subspace $S \subseteq Dim_{t(v)}$ is defined as: $N_S^V(v) = N_k^V(v) \cap N_{\epsilon,S}^V(v)$*

This neighborhood contains only vertices of the same type. This makes sense for the clustering as each cluster should only contain vertices of a single type. In Fig. 2, the combined neighborhood for $k = 2, \epsilon = 1$ of vertex A considering only dimension 1 would be $N_{\{1\}}^V(A) = \{A, B, C, D, E\}$. For dimension 2, $N_{\{2\}}^V(A) = \{A, C, D, E, F\}$.

Modeling Clusters and their Interactions. In our cluster definition, we have to ensure that all objects in a cluster are dense w.r.t. the combined neighborhood

[2] We choose the maximum norm because we want to consider two vertices similar only if they are similar in *all* of the dimensions in S.

[3] Another idea would be to combine the graph and attribute distance into a unified distance function. However, in that case a very small distance in the graph could even out a larger distance in the attribute space and vice versa. Instead, we want the vertices in the combined neighborhood to be similar in both regards.

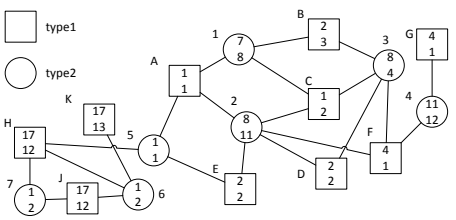


Fig. 2. Example with two vertex types

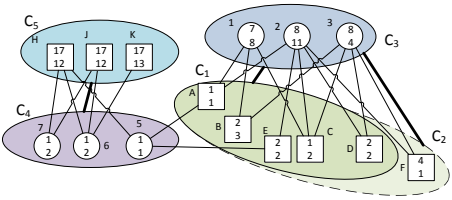


Fig. 3. Clustering for the example network

(ensured by property (1) in Def. 4) and the cluster is density-connected via the neighborhoods (property (2)). Furthermore, we want to detect the interactions between clusters of different types. Intuitively, two clusters of different types should be "connected" to each other if many edges exist between the vertices of these clusters. In other words, the connections to the vertices of the other cluster should induce density in a cluster. Therefore, we define a cluster C of vertices of type i *w.r.t. a set of clusters of the other types*. We call this set of clusters the *adjacent clusters* of C, denoted by $A(C)$. C has to be dense w.r.t. the union of the adjacent clusters. Therefore, the combined neighborhood is computed using only the vertices of the cluster itself and the union of the adjacent clusters:

Definition 4 (Typed Combined Subspace Cluster). *A typed combined subspace cluster $C = (O, S)$ of type i in a graph $G = (V, E, t, l)$ w.r.t. the parameters k_i, ϵ_i and $minPts_i$ and a set of adjacent clusters $A(C) = \{(O_j, S_j) \mid O_j \subseteq (V \setminus V_i), j = 1, \ldots, |A(C)|\}$ consists of a set of vertices $O \subseteq V_i$ and a set of relevant dimensions $S \subseteq Dim_i$[4] that fulfill the following properties:*

(1) density: $\forall v \in O : |N_S^{O \cup (\cup_{1 \leq j \leq a_C} O_j)}(v)| \geq minPts_i$
(2) connectivity: $\forall u, v \in O : \exists w_1, \ldots, w_l \in O : w_1 = u \wedge w_l = v \wedge \forall i \in \{1, \ldots, l-1\} : w_i \in N_S^{O \cup (\cup_{1 \leq j \leq a_C} O_j)}(w_{i+1})$
(3) density w.r.t. all adjacent clusters: $\forall (O_j, S_j) \in A : \exists W \subseteq O : (W, S)$ *forms a cluster w.r.t. the set* $A(W) = \{(O_j, S_j)\}$ *(for non-bipartite graphs: ignoring the edges* $(u, v) \in E : u, v \in O$*)*
(4) reciprocity: $\forall C_j \in A(C) : C \in A(C_j)$
(5) maximality: $\neg \exists O' \supset O : O'$ *fulfills (1) and (2)*

To avoid adding unrelated clusters, we require that each adjacent cluster has to induce density in at least a subset of C (property (3)). If there exist edges between vertices of the same type, we ignore them for testing this density (else, the cluster could be dense considering those edges alone, and thus the cluster definition would be fulfilled w.r.t. arbitrary other clusters). To avoid adding

[4] Generally, we require the subspace S to be non-empty, such that the vertices of the cluster show similarity in at least one dimension. However, in a heterogeneous network it is possible that not all of the vertex types have attributes. In this case, it makes sense to allow the detection of clusters with empty subspace. Thus, the user can choose if clusters with empty subspace should be included in the result.

clusters that just incidentally induce density in a small subset of C, we require a reciprocity of the adjacency between clusters (property (4)). Please note that we do *not* require a maximality property on $A(C)$ and redundant clusters can be removed from $A(C)$ later. Thus, a cluster C can fulfill the cluster definition for different sets $A(C)$. How to finally select $A(C)$ for the clustering result is discussed below.

The example network in Fig. 2 contains the following clusters (shown in Fig. 3) for the parameters $k_1 = k_2 = 2, \epsilon_1 = \epsilon_2 = 1, minPts_1 = minPts_2 = 3$:

- $C_1 = (\{A, B, C, D, E\}, \{1, 2\})$, connected to C_3
- $C_2 = (\{A, B, C, D, E, F\}, \{2\})$, connected to C_3
- $C_3 = (\{1, 2, 3\}, \{1\})$, connected to C_1, C_2
- $C_4 = (\{H, J, K\}, \{1, 2\})$, connected to C_5
- $C_5 = (\{5, 6, 7\}, \{1, 2\})$, connected to C_4

Interestingness of a TCSC Cluster. As we detect clusters in different subspaces, we can possibly find quite similar clusters in similar subspaces, like C_1 and C_2. To avoid redundancy in the result, we have to be able to decide which of the clusters is more interesting for our clustering result. Generally, we consider clusters with many vertices as interesting. However, a higher dimensionality also makes a cluster more interesting. Therefore, we introduce an interestingness function for clusters that considers both criteria. The interestingness function for a typed cluster is normalized by the overall number of vertices and the dimensionality of the corresponding type:

Definition 5 (Interestingness Measure). *The interestingness of a TCSC cluster $C = (O, S)$ of type i is defined as $Q(C) = \frac{|O| \cdot |S|}{|V_i| \cdot |Dim_i|}$ if $|Dim_i| > 0$, and $Q(C) = \frac{|O|}{|V_i|}$ else.*

In our example, $Q(C_1) = 0.35$ and $Q(C_2) = 0.3$. Thus, the two-dim. cluster C_1 is preferred as it is only slightly smaller than the similar one-dim. cluster C_2.

Parameters. Our model requires several parameters: ϵ, $minPts$ and k have to be set for each type. Setting these parameters for each type separately leads to a greater flexibility of the model: Especially if the number of vertices for the different types strongly differs, we should not expect the clusters of each type to fulfill e.g. the same $minPts$ value. In Section 5, we present a method for finding good parameter settings for ϵ and $minPts$. For k, a suitable setting can be directly obtained from the structure of the graph. E.g. for a bipartite paper-author network, $k_{author} = 2$ is a good choice as we need two hops to reach one author from another author. If there exist intra-type edges, we can consider them additionally by setting $k_{author} = 3$. For networks with more types, the distances between vertices of the same type can be larger, if vertices of different types have to be traversed. In this case, higher k values are required. Unlike other approaches, our model does not require the number of clusters as a parameter.

Selecting the Final Clustering Result. In order to avoid redundant clusters in the result, we first define a binary redundancy relation between two clusters of

the same type, adopting the definition from the DB-CSC model [7]. A cluster is considered redundant w.r.t. another cluster if its quality is lower and the clusters show a high overlap in their vertex sets as well as their relevant subspaces:

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

In Fig. 3, $C_2 \prec_{red} C_1$ (e.g. for $r_{obj} = r_{dim} = 0.5$).

Now we can define the final result set, which should not contain clusters that are redundant w.r.t. each other and has to be maximal w.r.t. this property. Furthermore, we ensure a maximality property for the set of adjacent clusters for a cluster C: If $C, A(C)$ together form a cluster according to Def. 4, then in the TCSC clustering the set of adjacent clusters of C must contain all the clusters in $A(C)$ except those that are redundant w.r.t. another cluster in $A(C)$.

Definition 7 (TCSC clustering). *Given the set Clusters of all TCSC clusters, the resulting TCSC clustering Result $\subseteq$ Clusters fulfills*

- *redundancy-freeness:* $\neg \exists C_i, C_j \in Result : C_i \prec_{red} C_j$
- *maximality:* $\forall C_i \in Clusters \setminus Result : \exists C_j \in Result : C_i \prec_{red} C_j$
- *maximality for adjacent clusters:* $\forall C : \forall C_i \in \{C_x \mid C \text{ fulfills Def. 4 w.r.t. } A(C) \cup \{C_x\}\} \setminus A(C) : \exists C_j \in A(C) : C_i \prec_{red} C_j$

Furthermore, we have to consider the connections between clusters of different types. A cluster $C \in Result$ may be adjacent to a cluster of another type that is excluded from the result due to redundancy. In this case, by just deleting this cluster we would lose the information about this connection. For solving this problem, we use the following theorem:

Theorem 1. *For the clustering defined in Def. 7 it holds: $\forall C \in Result : \forall C_A \in A(C) : (C_A \in Result \vee \exists C_{A'} \in Result : C_A \prec_{red} C_{A'})$.*

Proof Assume $\exists C \in Result, C_A \in A(C), C_A \in Clusters \setminus Result$. Following the maximality property in Def. 7 it holds $\exists C_{A'} \in Result : C_A \prec_{red} C_{A'}$.

I.e., if an adjacent cluster C_A is excluded from the result, there exists a similar cluster $C_{A'}$ that is contained in the result. In our implementation, we "reconnect" the cluster C to $C_{A'}$, if this connection does not yet exist. In Fig. 3, the result is $\{C_1, C_3, C_4, C_5\}$. □

4 The HSC Algorithm

In this section we give a short overview of the HSC algorithm. While HSC is partly based on DB-CSC [7], for heterogeneous networks novel challenges arise, which we discuss in this section. First, we explain the overall processing of the algorithm, followed by a detailed description of the refinement of a cluster.

```
method: main()
 1  Result = ∅ // current result set
 2  queue = ∅ // priority queue, sorted by quality
 3  for i = 1, ··· , T do A(V_i) = {V_j | 1 ≤ j ≤ T, j ≠ i}
 4  Detect set netclus of network-only clusters
 5  if network-only clusters are allowed then
 6      add all network clusters to queue
 7  for C ∈ netclus, d ∈ Dim_{t(O)} do DFS_trav(O, {d})
 8  repeat
 9      Sort queue ascendingly by dimensionality
10      for C = (O, S) ∈ queue : A(C) has changed do
11          refine_cluster(C)
12  until adjacency between clusters converges
13  while queue ≠ ∅ do
14      remove first cluster Clus from queue
15      if ∃C = (O, S) ∈ Result : Clus ≺_red C then
16          "reconnect" Clus's connections to C
17          goto line 13 // discard redundant cluster
18      Result = Result ∪ {Clus}
19  return Result
method: DFS_trav(vertex set O of type t, subspace S)
20  foundClusters = refine_cluster(C = (O, S))
21  add foundClusters to queue
22  for C_i = (O_i, S) ∈ foundClusters do
23      for d ∈ {max{S} + 1, ..., Dim_t} do
24          DFS_trav(O_i, S ∪ {d}) // check subsets of O_i
```

Algorithm 1. Pseudo-Code for the HSC algorithm

The pseudo-code for HSC is given in Algorithm 1. The final result set *Result* is initialized as an empty set (line 1), which is then filled iteratively by the algorithm until it contains the final, non-redundant clustering result defined in Def. 7. However, when a TCSC cluster C is detected during the processing, it can not directly be decided if C should be added to the clustering result, as a higher-quality cluster C' could be detected later such that $C \prec_{red} C'$ holds. Therefore, all detected clusters are temporarily stored in a priority queue *queue* (initialized in line 2) that is sorted according to the interestingness of the clusters.

From the definition of the combined neighborhood and the subspace distance, it follows that our TCSC clusters fulfill an anti-monotonicity property w.r.t. the subspaces, i.e. if there exists a cluster $C = (O, S)$ in subspace S, then for each subspace $S' \subset S$ there exists a vertex set $O' \supset O$ such that (O', S') also forms a TCSC cluster. This property can be exploited algorithmically: In order to detect clusters in higher-dimensional subspaces, the algorithm only has to check subsets of the vertex sets of the detected lower-dimensional clusters. Therefore, HSC starts by detecting clusters based only on the network information (i.e. clusters with the empty subspace). However, for heterogeneous networks, already the detection of these "network-only" clusters is a challenging problem. Because a cluster is defined based on its adjacent clusters of the other types, we can not simply determine the clusters for each type separately. Thus, the algorithm has to iteratively update the clusters and connections. A small example for the iterative processing on a paper-author network is illustrated in Fig. 4. If the

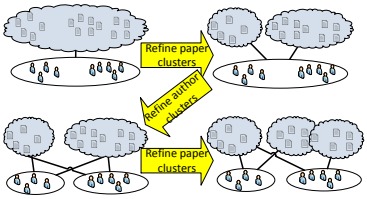


Fig. 4. Example for the detection of network clusters

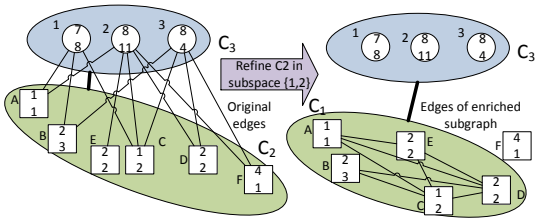


Fig. 5. Example for typed enriched subgraph

user allows clusters with empty subspace in the result, the network clusters are added to the *queue* (line 6). Based on the detected "network-only" clusters, HSC can now detect clusters in higher-dimensional subspaces (line 7). Based on these clusters, a depth-first search is performed in the subspaces[5] (line 20 – 24). Here, $max\{S\}$ denotes the dimension with maximal ID in subspace S.

Also for the higher-dimensional clusters, we encounter the challenge that the clusters of the different types depend on each other. During the DFS traversal, each cluster was refined based on the adjacent clusters that were currently known at the time of refinement. For many clusters in the queue, the set of adjacent clusters may have changed. Therefore, these clusters are refined based on the updated set of adjacent clusters (line 10 – 11). This process is repeated until the sets of adjacent clusters do not change anymore (line 12). For this step, the clusters are sorted ascendingly by dimensionality, as clusters of lower dimensionality tend to be connected to more clusters of the other types. After the set of clusters and adjacencies has converged, HSC detects the final clustering by processing the priority queue (sorted by quality), discarding clusters that are redundant w.r.t. a cluster already in the result set and adding non-redundant clusters to the result (line 18).

Refinement Based on Adjacent Clusters. Given a cluster candidate (i.e. a set of vertices O, a set of adjacent clusters and a subspace), the refinement method for the detection of TCSC clusters returns the set of all valid TCSC clusters with vertex sets $O' \subseteq O$ based on this subspace and adjacent clusters. The refinement method is based on a structure named *typed enriched subgraph*, which represents the similarity of the attributes in the considered subspace as well as the connectedness of the vertices via the adjacent clusters. In the typed enriched subgraph for a vertex set O, each vertex is connected to all vertices from its combined neighborhood (Def. 3), which is determined based on a given subspace S and using the connections via the vertices of O itself and of a vertex set O_A, which represents the vertices of all adjacent clusters:

[5] In practice, we can save computations and avoid the detection of some redundant clusters by using a technique from [7] that avoids the traversal of subspaces where probably only redundant clusters will be detected. Its details can be found in [7].

```
method: refine_cluster(cand. C = (O, S) of type t)
 1  foundClusters = ∅, prelimClusters = {C}
 2  for C_A ∈ A(C) do remove C from A(C_A)
 3  while prelimClusters ≠ ∅ do
 4      remove first C_x = (O_x, S) from prelimClusters
 5      compute adj. vertex set O_A = ∪_{(O_i,S_i)∈A(C_x)} O_i
 6      generate typed enriched subgraph G^{O_x}_{S,O_A}
 7      find (minPts_t − 1)-cores Cores = {O'_1, ..., O'_m}
 8      for each core O'_i determine adjacent clusters A(O'_i)
 9      if |Cores| = 1 ∧ O'_1 = O_x then
10          foundClusters = foundClusters ∪ {(O'_1, S)}
11      else prelimClusters = prelimClusters ∪ Cores
12  return foundClusters
```

Algorithm 2. Method for refining a single cluster

Definition 8 (Typed Enriched Subgraph). [6] *Given a set of vertices $O \subseteq V_i$, a subspace S, the original graph $G = (V, E, l)$ and a vertex set $O_A \subseteq V \setminus V_i$, the enriched subgraph $G^O_{S,O_A} = (V', E')$ w.r.t. O_A is defined by $V' = O$ and $E' = \{(u, v) \mid v \in N_S^{O \cup O_A}(u) \wedge v \neq u\}$ using the distance function $dist^S$.*

To fulfill the density property, each vertex in a TCSC cluster of type t has to have at least $minPts_t$ vertices in its combined neighborhood (which also contains the vertex itself). In the enriched subgraph, a TCSC cluster thus corresponds to a $(minPts_t - 1)$-core. In [7], it has been shown that the combined clusters can be detected by iteratively detecting $(minPts - 1)$-cores. Our method for finding TCSC clusters works in a similar fashion. The pseudo-code for the refinement method is given in Algorithm 2. If a candidate $C = (O, S)$ is refined, first the connection to C is removed from its adjacent clusters, which will later be connected to the new clusters detected in subsets of O (line 2). Then, HSC iteratively detects clusters in subsets of O (line 5 – 7). If O was not changed by the core-detection (line 9), the refinement has converged and the found vertex set is a cluster. However, if one or several smaller cores were detected, they cannot directly be output as clusters, because their adjacent clusters may have changed. Thus, the procedure is repeated (line 11) until convergence.

Fig. 5 shows an example for the graph from Fig. 2. Assume C_2 (in subspace $\{2\}$) and C_3 and their connection have already been detected. Now, we want to refine C_2 for the subspace $\{1, 2\}$. Thus, we construct the typed enriched subgraph for the vertices of C_2 based on this subspace and the vertices of C_3, and obtain the new cluster $\{A, B, C, D, E\}$ for the subspace $\{1, 2\}$.

5 Experiments

In this section we evaluate the performance of HSC (implemented in Java). We compare HSC to the algorithm of Tang et al. [17], the only existing algorithm for

[6] Please note that the typed enriched subgraph of a vertex set contains only vertices of the same type. However, it is determined using the combined neighborhood, which takes the connections to the adjacent clusters into account.

clustering heterogeneous networks that also separates vertices of different types into different clusters and (in its extension) considers vertex attributes.[7]

Data Sets. For our experiments, we use two heterogeneous real-world data sets that each contain several vertex attributes. Yelp is a website where users can rate and review businesses. Our yelp network was extracted from the yelp academic data set (http://www.yelp.com) and has three vertex types: "User", "Business" and "Review". The network contains all the businesses from the academic data set that belong to the categories "Restaurant" and "Pizza", all the users who rated at least one of these businesses and all of the corresponding reviews. Overall, there are 6931 user vertices, 283 business vertices and 8584 review vertices. A review vertex is connected to the user who submitted this review as well as to the business it rates. Furthermore, the network also contains intra-type edges and thus is not tripartite: Two businesses are connected by an edge if they are located close to each other (up to 300m apart). For all vertex types, the data set provides additional attribute information: For the businesses, we have the attributes "review count" (number of ratings received) and "average rating" (from 1 to 5 stars). For the users, we also have "review count" (number of ratings submitted) and "average rating" (of the ratings by this user). Furthermore, user vertices have the attributes "funny", "useful" and "cool", which correspond to attributes given to the reviews of a user by other users. Review vertices have the single attribute "stars". All values were normalized to $[0, 1]$.

Our second network was extracted from the DBLP (http://dblp.uni-trier.de) database and has the vertex types "Author" and "Paper". It contains all papers of selected conferences from the database and data mining area from the years 2000 to 2004. Each paper is connected to the vertices representing its authors. Authors vertices do not contain attributes. The papers have binary attributes indicating the occurrence of certain keywords in the title. To avoid irrelevant keywords, only words that occurred in at least 100 papers are represented. Overall, we have 5497 author vertices and 3354 paper vertices with 208 attributes.

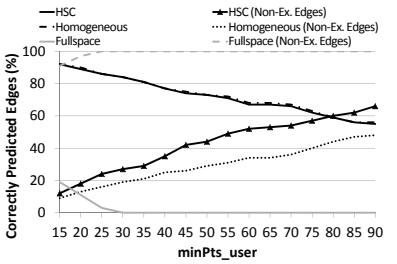
Experiments on Yelp Data. As no ground truth for the clustering is available, we can not evaluate the clustering quality directly. Therefore, we divide the edges of the Yelp data set into a training set (95% of the edges) and a test set (5% of the edges). Using these data sets, we obtain an accuracy measure that measures how well the test edges are predicted by the result of HSC, i.e. which percentage of the test edges connect vertices from adjacent clusters in our result. We also create a test set of edges between vertices that are not connected in the network and use this set to obtain a "false positive" rate. Please note that we can not expect to reach perfect or nearly perfect accuracy values using this measure, as this would only be possible for graphs where the clusters correspond to fully connected cliques and no edges between clusters exist, which does not hold for real graphs. However, we can use this measure to compare how well different clustering results capture the structure of the graph. Using this method, we analyze the influence

[7] We extended the implementation from the authors homepage for attribute values as described in [17] and treat our networks as networks with a single time stamp.

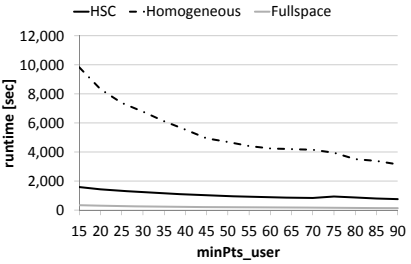
of parameters on the result. In each experiment, we measure the percentage of correctly predicted edges and of correctly predicted non-existing edges. This method can be used for finding good parameter setting. For each parameter, we choose the value maximizing the minimum of both accuracy values. For HSC, we vary the values for ϵ and $minPts$ for each type. The parameter k is discrete and is set as discussed in Section 3. For the method from [17], the number of clusters for each type has to be given as a parameter, thus we vary these values. However, for this method computing the accuracy is problematic, as it does not produce binary connections between clusters. Instead, we consider the group interaction matrix A that is used by [17] and interpret positive entries as "connection" and negative entries as "no connection".

To analyze the advantage of our heterogeneous clustering method over clustering methods for homogeneous graphs, we test a baseline that projects the heterogeneous network to one homogeneous network for each vertex type (e.g. connecting two users if they are connected to the same business in the heterogeneous network) and then uses DB-CSC [7] on each network separately. To enable a comparison with the results of HSC, this baseline detects connections between the clusters from the different networks in a post-processing step. To analyze the influence of subspace clustering on our results, we also test a fullspace version of HSC that detects only clusters that show similarity in all dimensions.

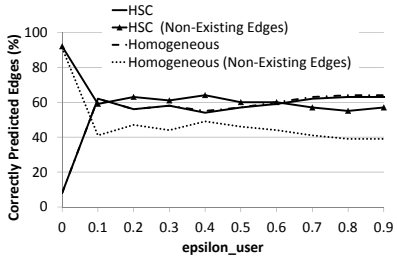
The results for the "user" vertex type are shown in Fig. 6. The experiments for the other vertex types show similar results (not printed here due to space limitations). For the parameter $minPts_{user}$, increasing values lead to a lower accuracy for the test edges and a higher accuracy for the non-existing test edges (Fig. 6(a)). This is due to the fact that for higher $minPts$ values, less and smaller TCSC clusters are detected due to the stricter density criterion. Therefore, less correct edges, but also less non-existing edges are predicted by the clustering result. According to this results, we set $minPts_{user} = 50$. The baseline clustering the homogeneous projections of the network separately reaches very similar values in the accuracy for test edges. However, the accuracy for non-existing test edges is considerably worse than for HSC. This is due to the fact that this method cannot use the information about the clustering structure of the other vertex type, and thus also clusters vertices together that are connected via noise vertices of the other type. Therefore, the resulting clusters are supersets of the TCSC clusters and the predicted connections show worse accuracy for non-existing test edges. We also evaluate the fullspace version of HSC in this experiment. We observe that only few clusters can be found in the fullspace and thus the accuracy value for existing edges is very low. For values greater than 25, the fullspace version detects no clusters at all. Therefore, the fullspace version is not used in the following experiments. The runtimes for all versions (Fig. 6(b)) decrease for increasing $minPts$ values, as less clusters are detected. The runtimes of HSC are slightly higher than those of the fullspace version of HSC, as the fullspace version does not have to look for clusters in different subspaces. The method clustering homogeneous networks shows far higher runtimes, as it



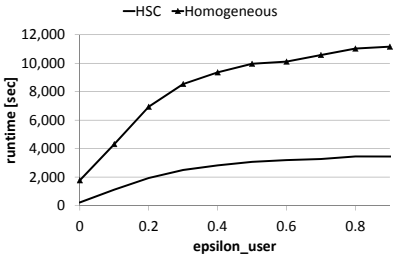
(a) Accuracy vs. $minPts_{user}$ for HSC



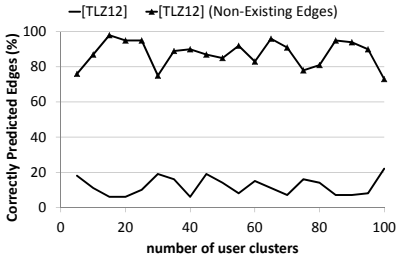
(b) Runtime vs. $minPts_{user}$ for HSC



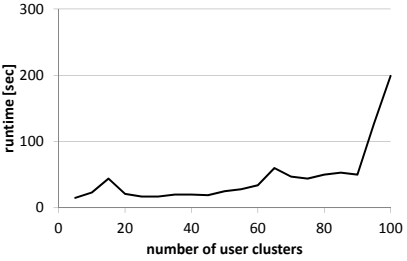
(c) Accuracy vs. ϵ_{user} for HSC



(d) Runtime vs. ϵ_{user} for HSC



(e) Accuracy vs. #user clusters for [17]



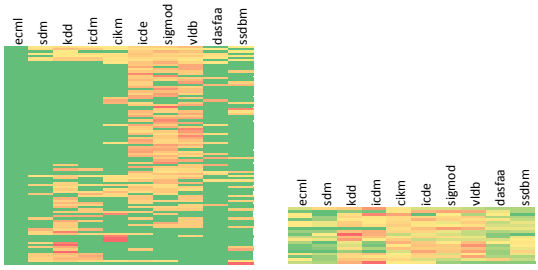
(f) Runtime vs. #user clusters for [17]

Fig. 6. Experimental results on the Yelp data set with 3 vertex types

has to cluster all networks separately and cannot use the information about the clustering structure of the respective other vertex types for pruning.

For an increasing ϵ_{user}, both accuracy values for HSC remain relatively stable (Fig. 6(c)). Like in the previous experiment, the homogeneous clustering variant shows similar behavior in the accuracy for test edges and considerably worse values for the accuracy for non-existing test edges. The runtime (Fig. 6(d)) increases quickly for increasing ϵ-values until it reaches a plateau, as for higher ϵ-values larger vertex neighborhoods have to be considered. Again, the runtime of the homogeneous variant is far higher than that of HSC.

For the method from [17], we do not observe a trend in the accuracy for an increasing number of user clusters (Fig. 6(e)). In contrast to HSC, in this partitioning method each vertex is grouped in exactly one cluster, thus the trend described above does not occur here. The accuracy for test edges is considerably lower than that of HSC, while a high accuracy for non-existing edges is obtained.



(a) HSC (b) Method from [17]

Fig. 7. Distribution of paper clusters over conferences for the DBLP data set (heat map color gradient from Green="0%" to Red="100% of the papers in the cluster belong to this conference")

This shows that this approach predicts less connections between clusters for the 3-type network. Overall, the method shows far lower runtimes than HSC (Fig. 6(f)), as it does not consider subspaces of the attribute space and does not exclude outliers. However, HSC reaches better accuracy values: We can find parameter settings such that both accuracy values for HSC are about 60%.

Experiments on DBLP Data. On DBLP, HSC detects 14 author clusters with an average size of 59 and 98 paper clusters with an average size of 10 and average dimensionality of 1.05, i.e. most of the paper clusters have one or two keywords in common. The method from [17] detects 20 paper clusters with an average size of 168 and 20 author clusters with an average size of 275. As this method does not consider subspace clusters, there is no information about the relevant keywords for the clusters; the papers in a cluster do not necessarily have common keywords at all. However, considering subspaces and connections between clusters also causes higher runtimes: The runtime was 5 sec. for the method from [17] and 278 sec. for HSC. To provide an impression of the detected clusters, we depict the distribution of the detected paper clusters over the conferences in Fig. 7. Each row in the diagrams corresponds to a paper cluster. For most of the clusters detected by HSC, the papers in the cluster belong to a small set of conferences. Particularly, most clusters show a clear tendency either to the database area or the data mining area. For the method from [17], the tendency is less clear.

6 Conclusion

We propose the clustering model TCSC for the clustering of vertices in heterogeneous networks, which takes into account the connections in the network as well as the vertex attributes. Furthermore, TCSC detects interactions between clusters of different types. TCSC is the first clustering model which considers subspace clustering in heterogeneous networks. We introduce the algorithm HSC for computing the TCSC clustering.

Acknowledgements. This work has been supported by the B-IT Research School of the Bonn-Aachen International Center for Information Technology.

References

1. Akoglu, L., Tong, H., Meeder, B., Faloutsos, C.: Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In: Proceedings of the Twelfth SIAM International Conference on Data Mining, pp. 439–450 (2012)
2. Barber, M.: Modularity and community detection in bipartite networks. Phys. Rev. E 76(6), 066102 (2007)
3. Boden, B.: Combined Clustering of Graph and Attribute Data. Ph.D. thesis, RWTH Aachen University, Aachen (2014)
4. Boden, B., Günnemann, S., Hoffmann, H., Seidl, T.: Mining coherent subgraphs in multi-layer graphs with edge labels. In: SIGKDD, pp. 1258–1266 (2012)
5. Gao, B., Liu, T., Zheng, X., Cheng, Q., Ma, W.: Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In: SIGKDD, pp. 41–50 (2005)
6. Günnemann, S., Färber, I., Boden, B., Seidl, T.: Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In: ICDM (2010)
7. Günnemann, S., Boden, B., Seidl, T.: Finding density-based subspace clusters in graphs with feature vectors. DMKD 25(2), 243–269 (2012)
8. 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)
9. Li, X., Ng, M.K., Ye, Y.: Multicomm: Finding community structure in multi-dimensional networks. TKDE 99(PrePrints), 1 (2013)
10. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM, pp. 593–604 (2009)
11. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.P.: Community structure in time-dependent, multiscale, and multiplex networks. Science 328(5980), 876–878 (2010)
12. Shiga, M., Takigawa, I., Mamitsuka, H.: A spectral clustering approach to optimally combining numerical vectors with a modular network. In: SIGKDD, pp. 647–656 (2007)
13. Sun, Y., Aggarwal, C., Han, J.: Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. VLDB 5(5), 394–405 (2012)
14. Sun, Y., Norick, B., Han, J., Yan, X., Yu, P., Yu, X.: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In: SIGKDD, pp. 1348–1356 (2012)
15. Sun, Y., Yu, Y., Han, J.: Ranking-based clustering of heterogeneous information networks with star network schema. In: SIGKDD, pp. 797–806 (2009)
16. Suzuki, K., Wakita, K.: Extracting multi-facet community structure from bipartite networks. In: CSE, vol. 4, pp. 312–319 (2009)
17. Tang, L., Liu, H., Zhang, J.: Identifying evolving groups in dynamic multimode networks. TKDE 24(1), 72–85 (2012)
18. Tang, L., Wang, X., Liu, H.: Community detection via heterogeneous interaction analysis. DMKD 25(1), 1–33 (2012)
19. Yang, J., McAuley, J.J., Leskovec, J.: Community detection in networks with node attributes. In: ICDM, pp. 1151–1156 (2013)
20. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB 2(1), 718–729 (2009)